Chapter 7

Deploying Multi-Container Applications

Considerations for Multi-Container Applications

Managing Multi-Container Applications

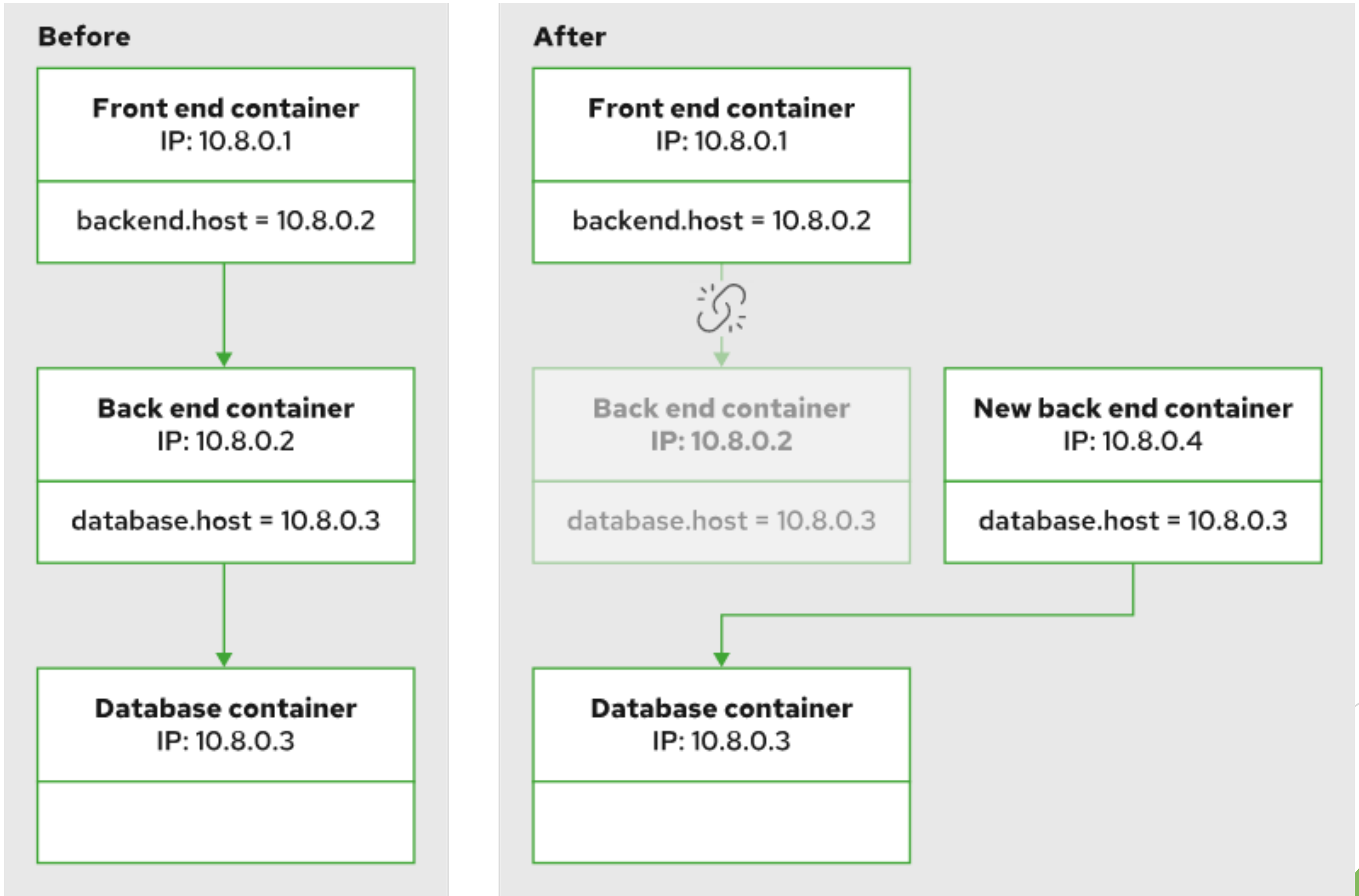
- Consider an application composed of a front-end web application, a REST back end, and a database server.
- Although it is possible to **orchestrate** multi-container applications' containers manually, Kubernetes and OpenShift provide tools to facilitate orchestration.
- We are going to return to using **Podman** to create a simple multi-container application to demonstrate the underlying manual steps for container orchestration.
- In later sections, you will use **Kubernetes and OpenShift** to orchestrate these same application containers.

Discovering Services in a Multi-Container Application

- Container Network Interface (CNI) assigns a new IP address to a container when it starts.
- Each container exposes all ports to other containers in the same SDN.
- The containers expose ports to external networks only by explicit configuration.
- Due to the dynamic nature of container IP addresses, applications cannot rely on either fixed IP addresses or fixed DNS host names to communicate with middleware services and other application services.

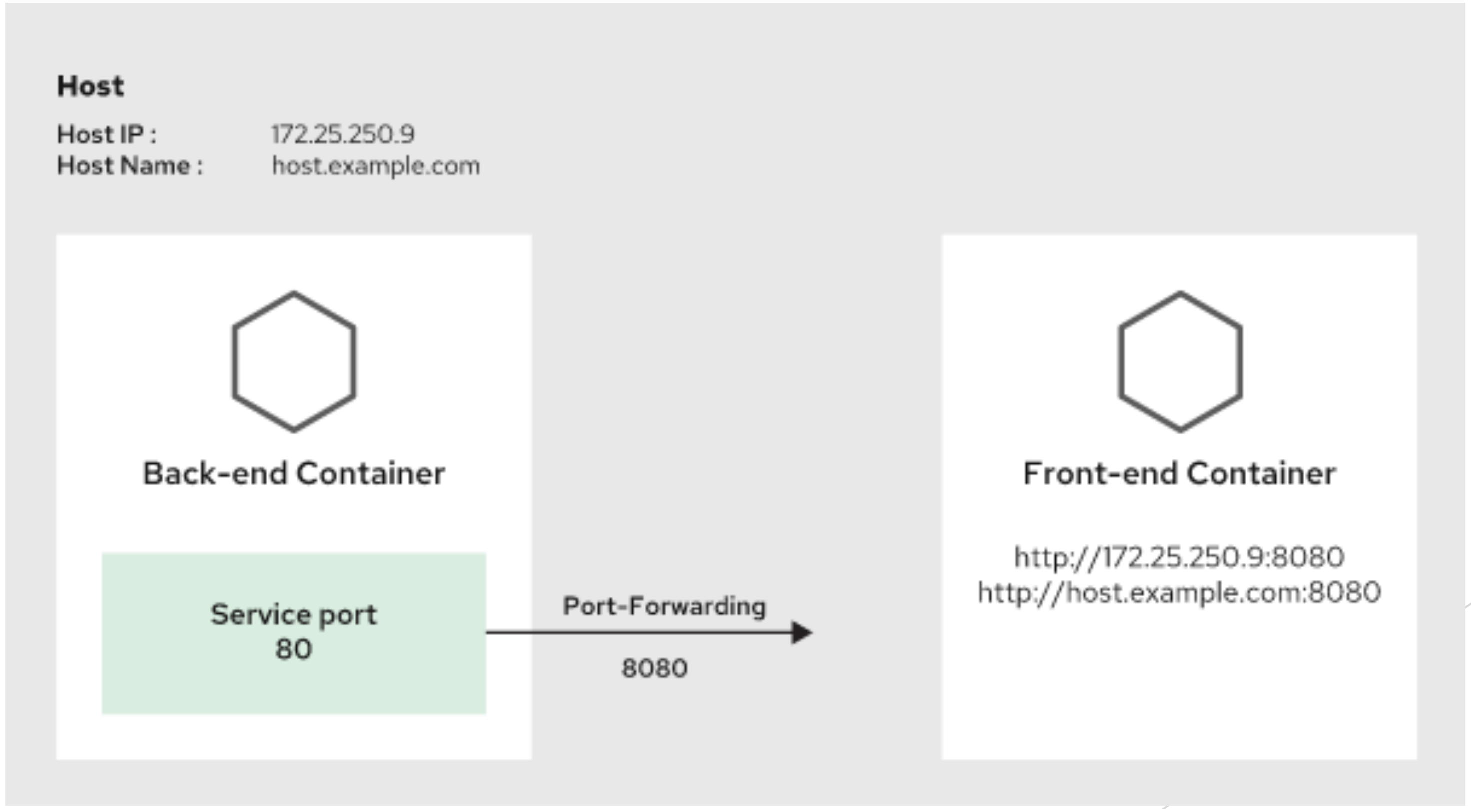
Rootfull Containers

A restart breaks three-tiered application links

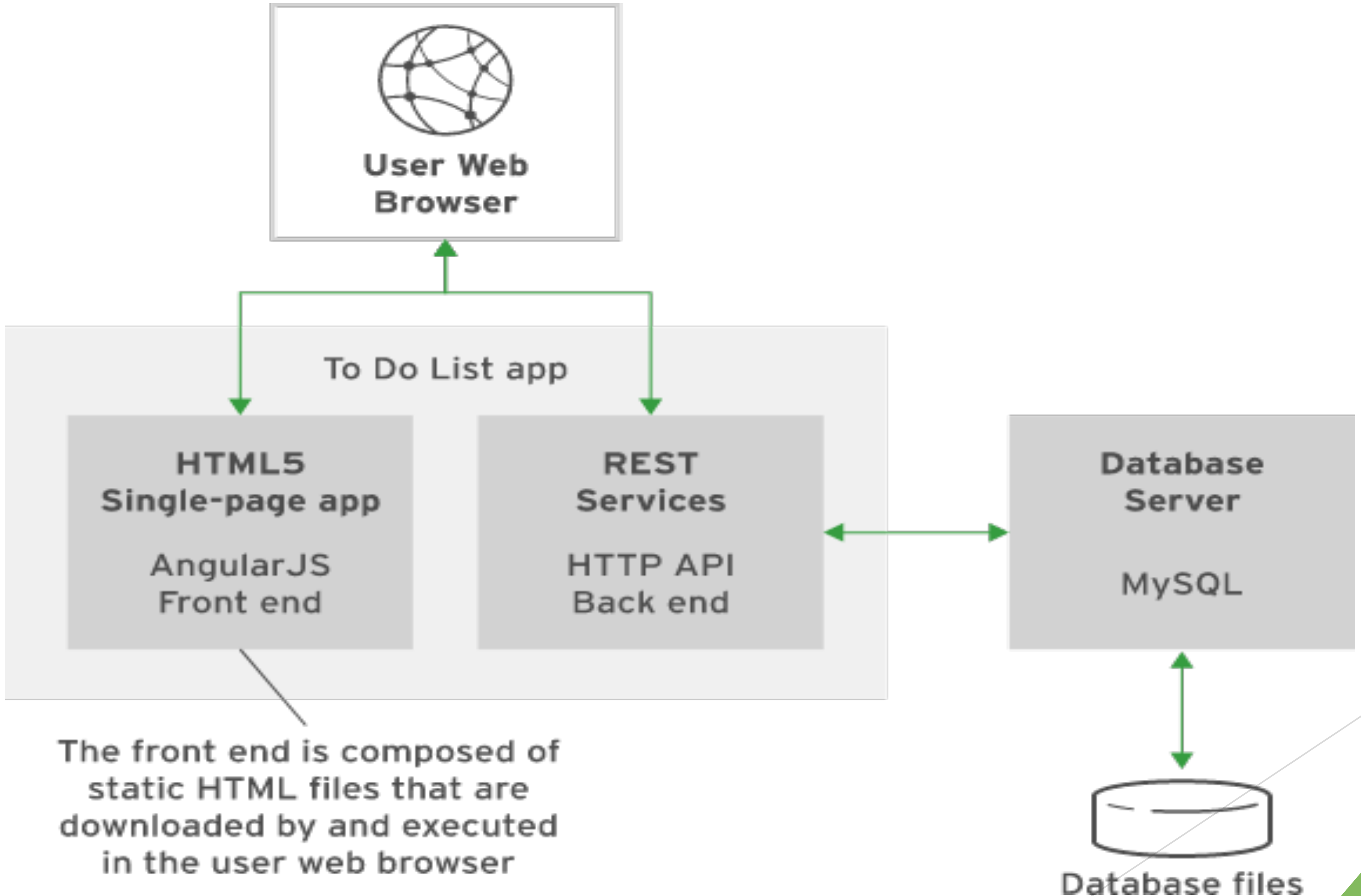


Rootless Containers

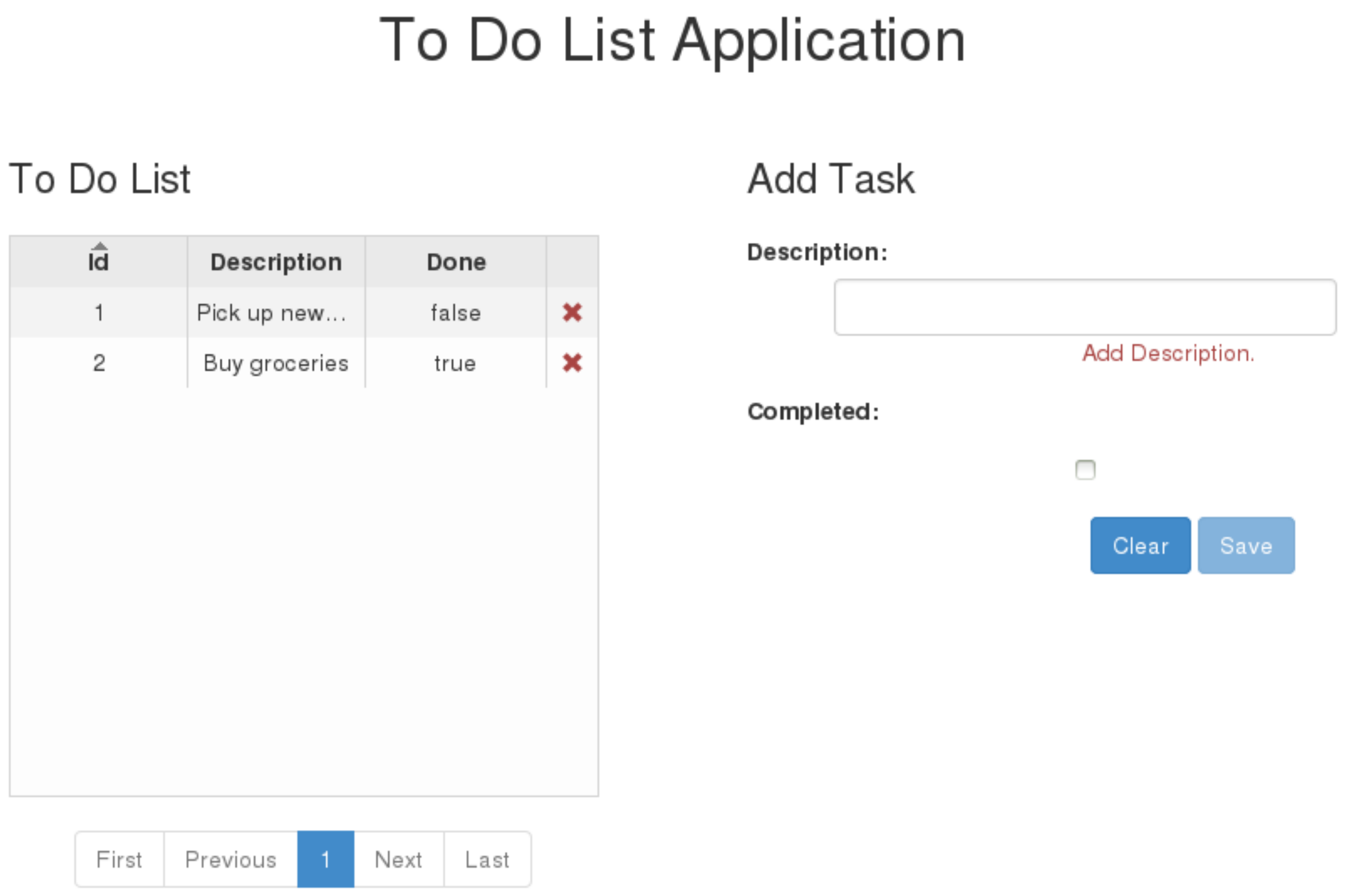
- Networking between rootless containers by using port-forwarding.
- Port-forwarding allows external access to a container service from the host.



Describing the To Do List Application



Todo Application Web Interface



To Do List Application

To Do List

Id	Description	Done	
1	Pick up new…	false	✖
2	Buy groceries	true	✖

Add Task

Description:

Add Description.

Completed:

Guided Exercise: Deploying the Web Application and MySQL on Linux Containers

- https://rha.ole.redhat.com/rha/app/courses/do180-4.10/52f11f0e-a277-4441-9d11-e3d56d7defca/pages/ch07s02

Deploying a Multi-Container Application on OpenShift

Comparing Podman and Kubernetes

- Using environment variables allows you to share information between containers with Podman.
- However, there are still some limitations and some manual work involved in ensuring that all environment variables stay in sync, especially when working with many containers.
- Kubernetes provides an approach to solve this problem by creating services for your containers, as covered in previous chapters.

Services in Kubernetes

- Pods are attached to a Kubernetes namespace, which OpenShift calls a *project*.
- When a pod starts, Kubernetes automatically adds a set of environment variables for each service defined on the same namespace.
- Any service defined on Kubernetes generates environment variables for the IP address and port number where the service is available.
- Kubernetes automatically injects these environment variables into the containers from pods in the same namespace.

An Example

given the following service:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    name: mysql
  name: mysql
spec:
  ports:
    - protocol: TCP
    - port: 3306
  selector:
    name: mysql
```

The following environment variables are available for each pod created after the service, on the same namespace:

```
MYSQL_SERVICE_HOST=10.0.0.11
MYSQL_SERVICE_PORT=3306
MYSQL_PORT=tcp://10.0.0.11:3306
MYSQL_PORT_3306_TCP=tcp://10.0.0.11:3306
MYSQL_PORT_3306_TCP_PROTO=tcp
MYSQL_PORT_3306_TCP_PORT=3306
MYSQL_PORT_3306_TCP_ADDR=10.0.0.11
```

Guided Exercise: Creating an Application on OpenShift

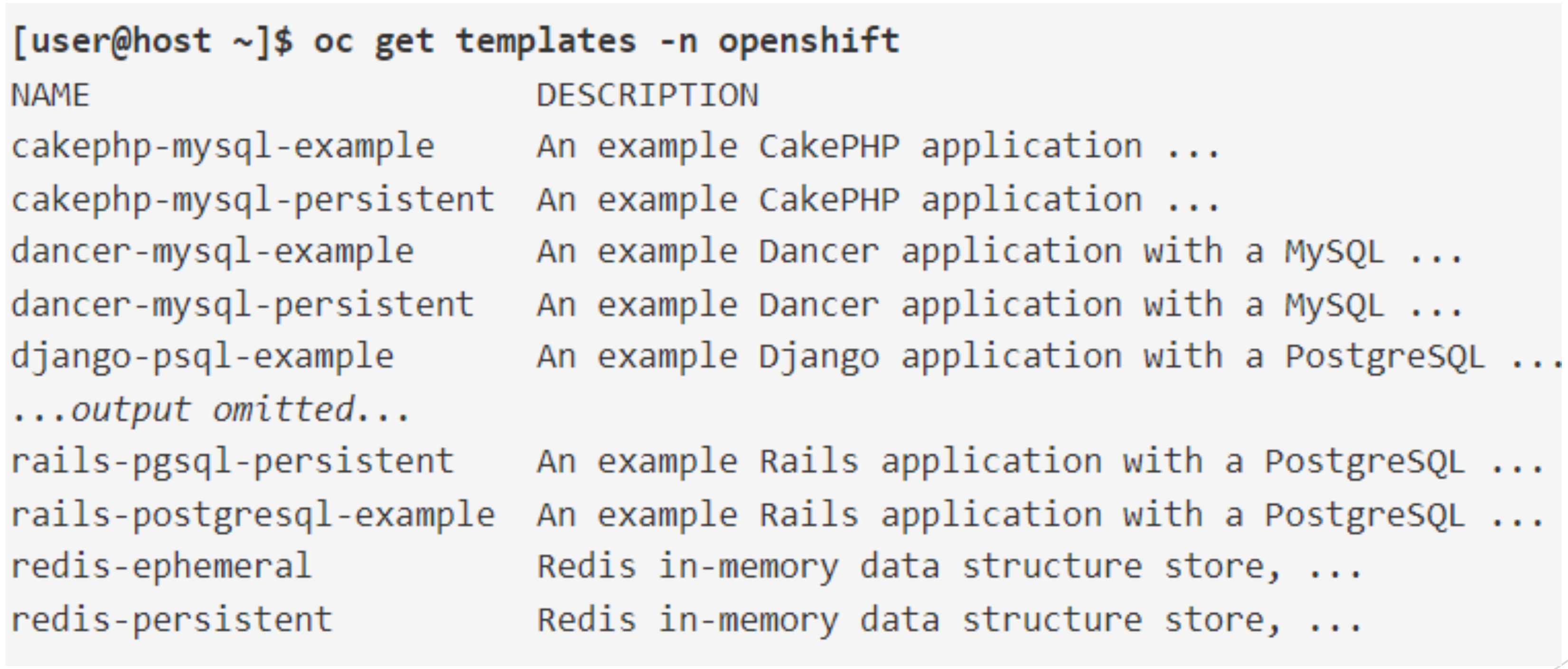
- https://rha.ole.redhat.com/rha/app/courses/do180-4.10/52f11f0e-a277-4441-9d11-e3d56d7defca/pages/ch07s04

Deploying a Multi-container Application on OpenShift Using a Template

What is a template?

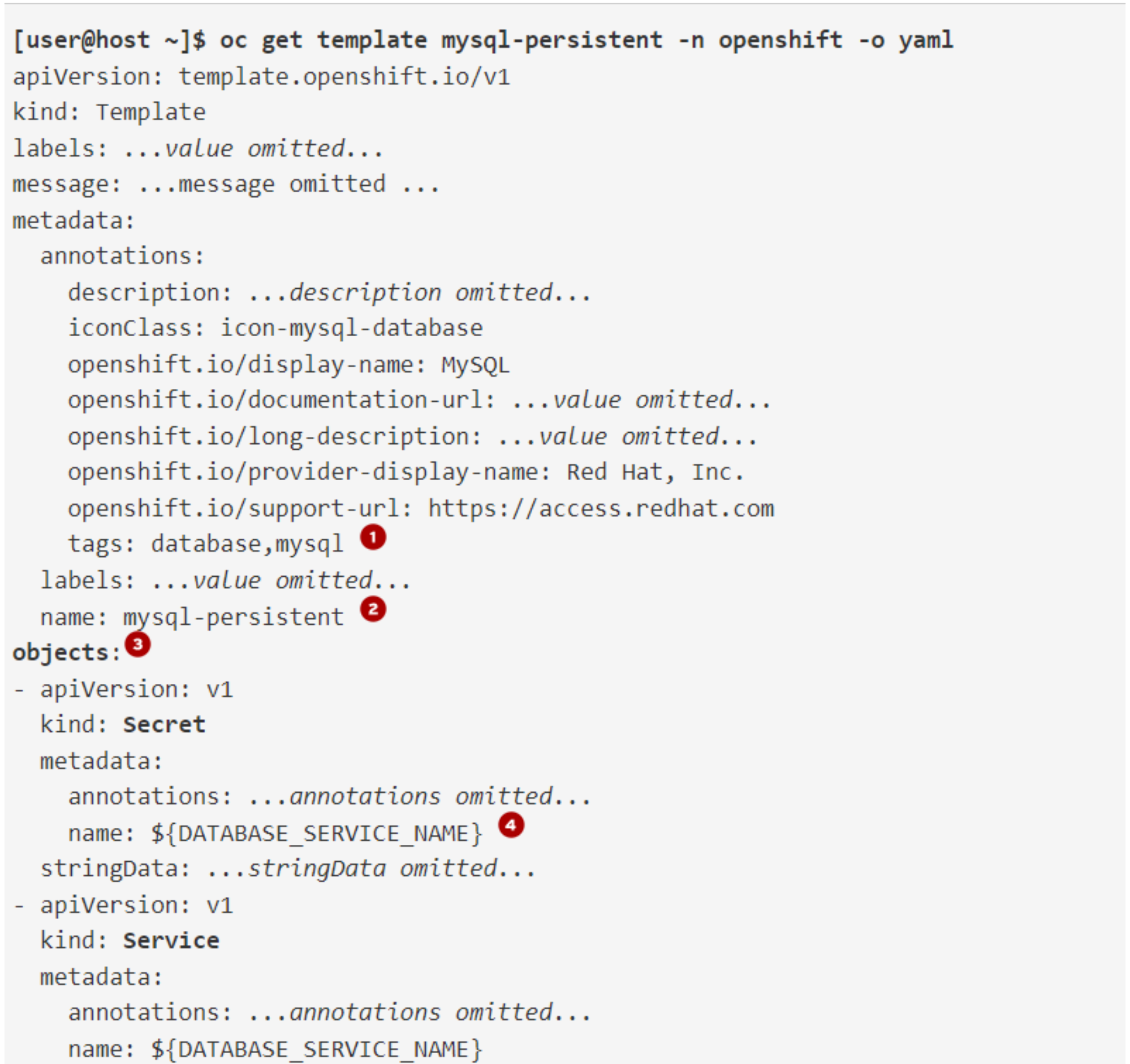
- Deploying an application on OpenShift Container Platform often requires creating several related resources within a Project.
- OpenShift templates provide a way to simplify the creation of resources that an application requires. A template defines a set of related resources to be created together, as well as a set of application parameters.
- The attributes of template resources are typically defined in terms of the template parameters, such as a resource’s name attribute.
- For example, an application might consist of a front-end web application and a database server. Each consists of a service resource and a deployment resource. They share a set of credentials (parameters) for the front end to authenticate to the back end.
- The template can be processed by specifying parameters or by allowing them to be automatically generated (for example, for a unique database password) in order to instantiate the list of resources in the template as a cohesive application.

OpenShift built in templates



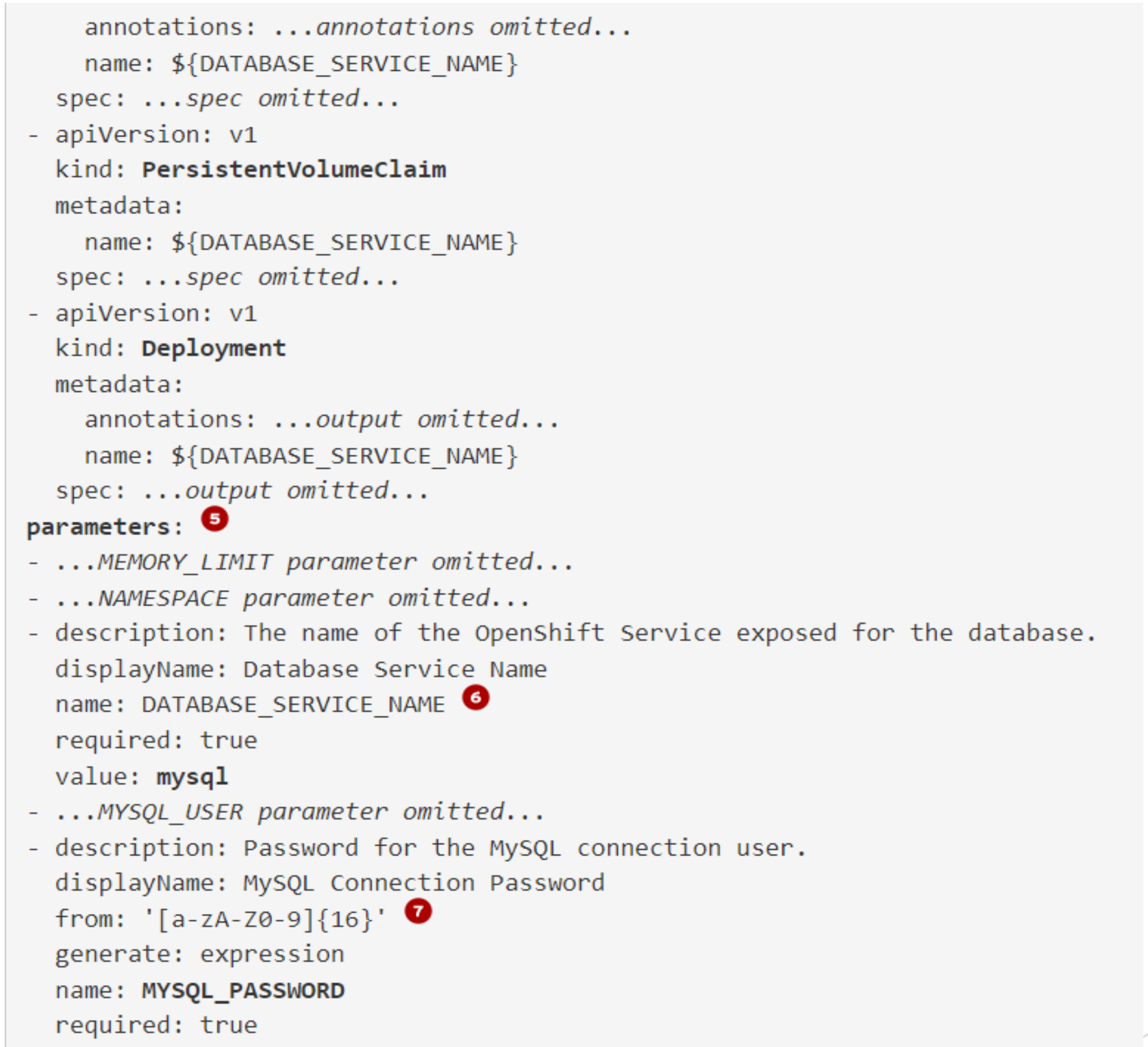
```
[user@host ~]$ oc get templates -n openshift
NAME                        DESCRIPTION
cakephp-mysql-example       An example CakePHP application ...
cakephp-mysql-persistent    An example CakePHP application ...
dancer-mysql-example        An example Dancer application with a MySQL ...
dancer-mysql-persistent     An example Dancer application with a MySQL ...
django-psql-example         An example Django application with a PostgreSQL ...
...output omitted...
rails-pgsql-persistent      An example Rails application with a PostgreSQL ...
rails-postgresql-example    An example Rails application with a PostgreSQL ...
redis-ephemeral             Redis in-memory data structure store, ...
redis-persistent            Redis in-memory data structure store, ...
```

Show and edit a template yaml file



```
[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}
```

[cont.] Show and edit a template yaml file



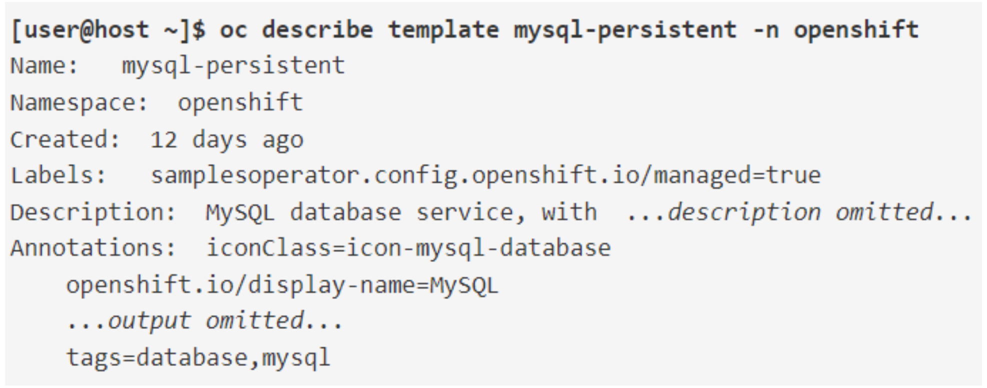
```
    annotations: ...annotations omitted...
    name: ${DATABASE_SERVICE_NAME}
  spec: ...spec omitted...
- apiVersion: v1
  kind: PersistentVolumeClaim
  metadata:
    name: ${DATABASE_SERVICE_NAME}
  spec: ...spec omitted...
- apiVersion: v1
  kind: Deployment
  metadata:
    annotations: ...output 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
```

Parameters

- Templates define a set of parameters, which are assigned values. OpenShift resources defined in the template can get their configuration values by referencing named parameters.
- Parameters in a template can have default values, but they are optional. Any default value can be replaced when processing the template.
- Each parameter value can be set either explicitly by using the oc process command, or generated by OpenShift according to the parameter configuration.

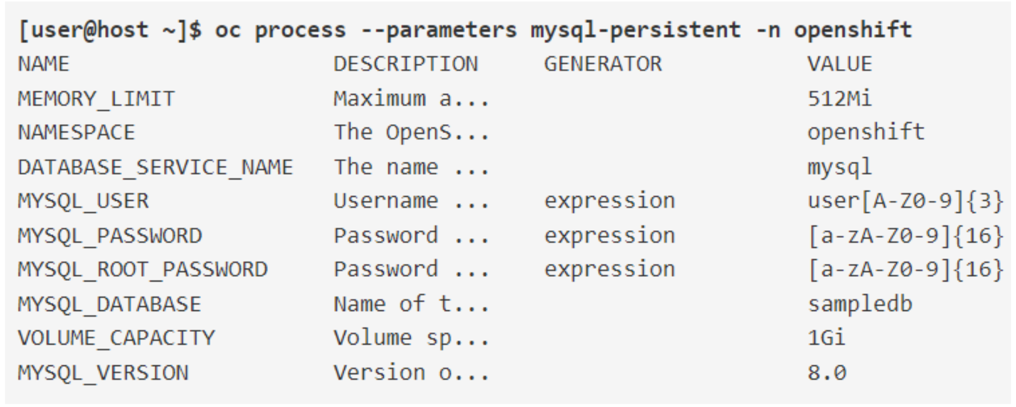
Parameters

- There are two ways to list available parameters from a template.
- The first one is using the *oc describe* command:



```
[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
```

- The second way is by using the *oc process* with the *--parameters* option:



```
[user@host ~]$ oc process --parameters mysql-persistent -n openshift
NAME                    DESCRIPTION     GENERATOR       VALUE
MEMORY_LIMIT            Maximum a...                    512Mi
NAMESPACE               The OpenS...                    openshift
DATABASE_SERVICE_NAME   The name ...                    mysql
MYSQL_USER              Username ...    expression      user[A-Z0-9]{3}
MYSQL_PASSWORD          Password ...    expression      [a-zA-Z0-9]{16}
MYSQL_ROOT_PASSWORD     Password ...    expression      [a-zA-Z0-9]{16}
MYSQL_DATABASE          Name of t...                    sampledb
VOLUME_CAPACITY         Volume sp...                    1Gi
MYSQL_VERSION           Version o...                    8.0
```

Processing a Template Using the CLI

- See Material for extra commands

Guided Exercise: Creating an Application with a Template

- https://rha.ole.redhat.com/rha/app/courses/do180-4.10/52f11f0e-a277-4441-9d11-e3d56d7defca/pages/ch07s06

Lab: Deploying Multi-Container Applications

- https://rha.ole.redhat.com/rha/app/courses/do180-4.10/52f11f0e-a277-4441-9d11-e3d56d7defca/pages/ch07s07