Deploying Multi-Container Applications

Considerations for Multi-Container Applications

After completing this sections, students should be able to:

Describe considerations for containerizing applications with multiple container image

Leverage networking concepts in containers

Create a multi-container application with Podman

Describe the architecture of the To Do List application

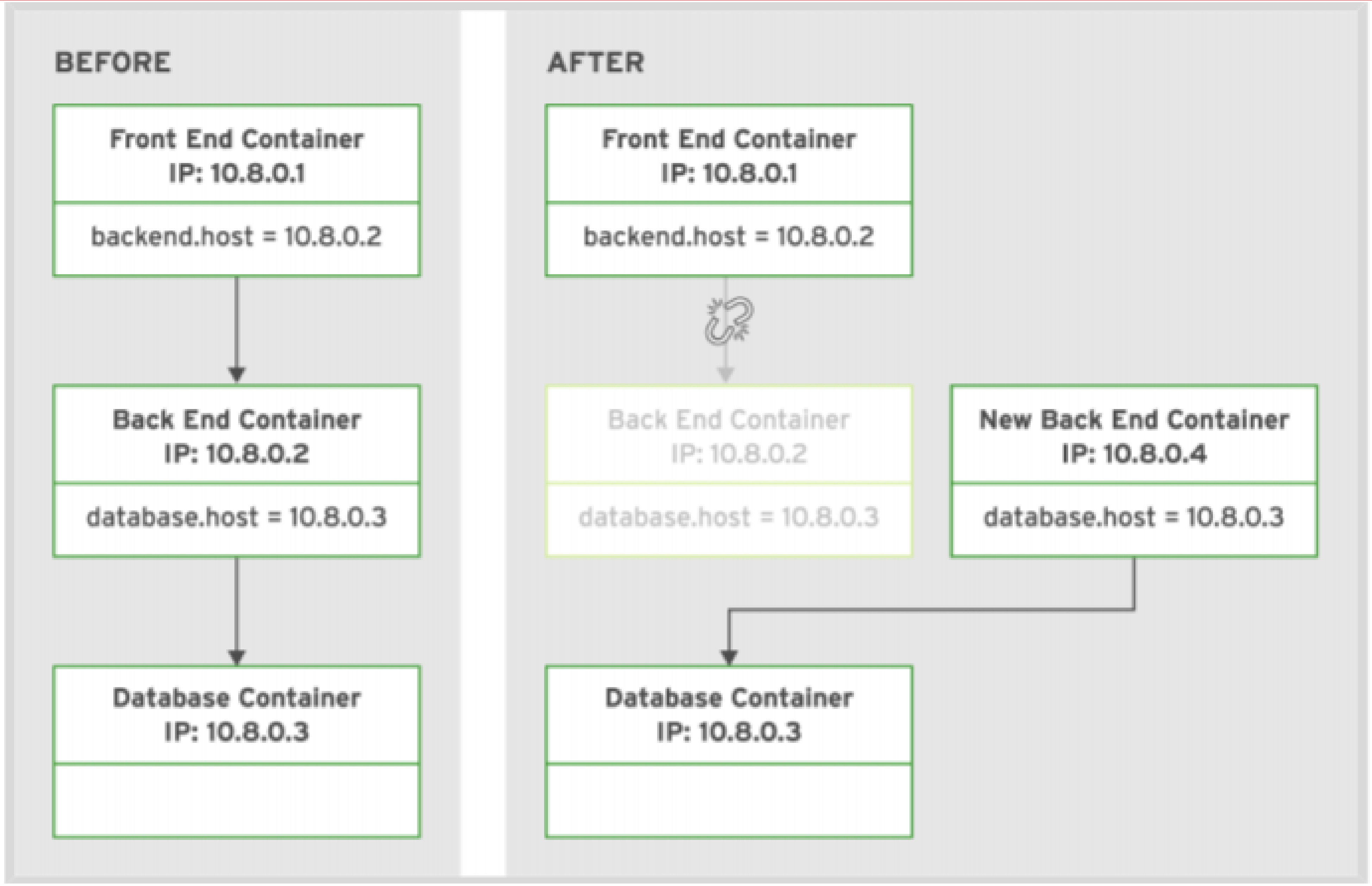
Leveraging Multi-Container Applications

- Common setup
- Conform to multi-tier applications
- Example:
  - ❑ Front-end : Web application
  - ❑ Back-end: Database server
- Each have different dependencies, requirements and life cycles
- Manual management becomes complicated
- Orchestrate with Kubernetes or OpenShift

Discovering Services in a Multi-Container Application

- Rootfull containers vs Rootless containers
- Podman uses CNI to SDN
- SDN provides communication between pods
- CNI assigns new IP address to container
- Due to dynamic nature of container IP, it’s a challenge for multi-tier containerized application communicate with each other

A restart breaks three-tiered application links



Comparing Podman and K8s

- Use of environment variables, information can be shared among containers in Podman
- Become complex when manage large number of containers in Podman
- K8s solved this by creating namespace and deployments
- Pods are attached to namespace and deployments.
- Services defined generates environment variables for IP address and port number
- K8s add this environment variable to PODs as they start
- Standard environment variables use following convention:
    - ❑ Uppercase: All env. Variables set using uppercase
    - ❑ Snakecase: all spaces separated by underscore (_)
    - ❑ Service name first:
    - ❑ Protocol type: Most network protocol either use TCP or UDP

Environment Variables generated by K8s for service

<SERVICE_NAME>_SERVICE_HOST

Ip address enabled by service to access a pod

<SERVICE_NAME>_SERVICE_PORT

Address, port and protocol provided by service for external access

<SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>

Define an alias for the <SERVICE_NAME>_PORT

<SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>_PROTO

Identifies protocol type (UDP or TCP)

<SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>_PORT

Define an alias for the <SERVICE_NAME>_SERVICE_PORT

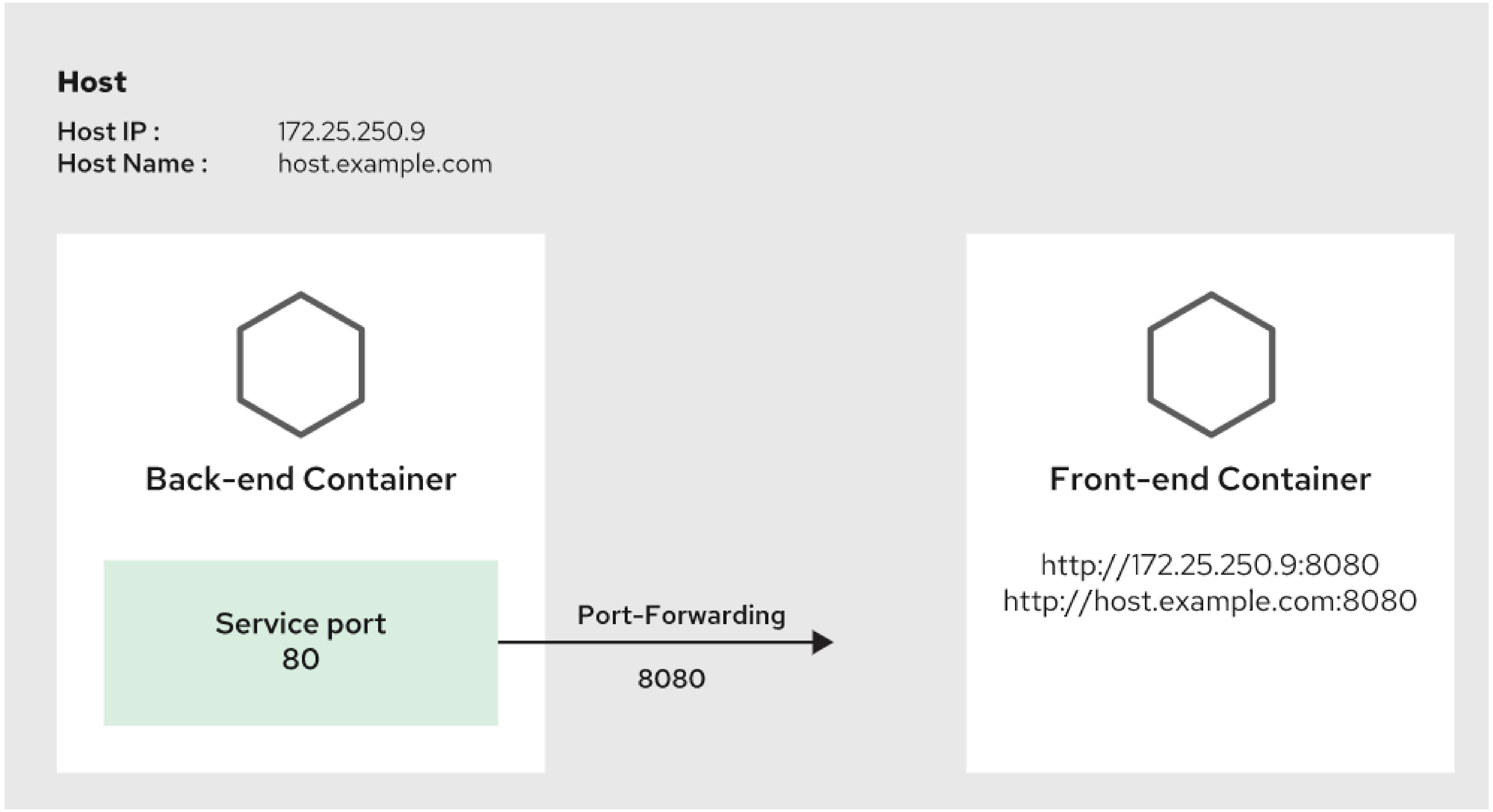
…

Environment Variables generated by K8s for service

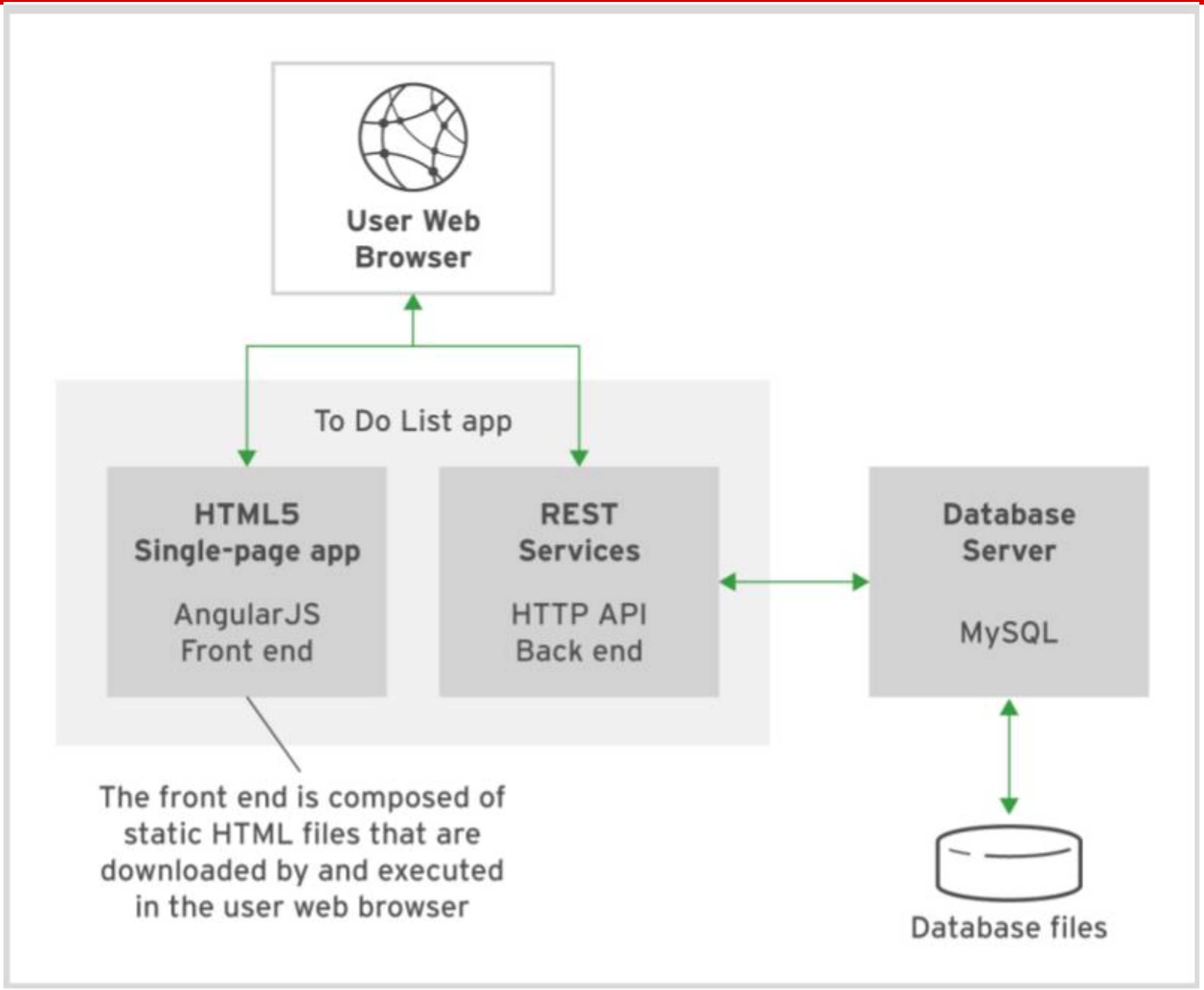
```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    name: mysql
  name: mysql
spec:
  ports:
    - protocol: TCP
    - port: 3306
  selector:
    name: mysql
```

```
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
```

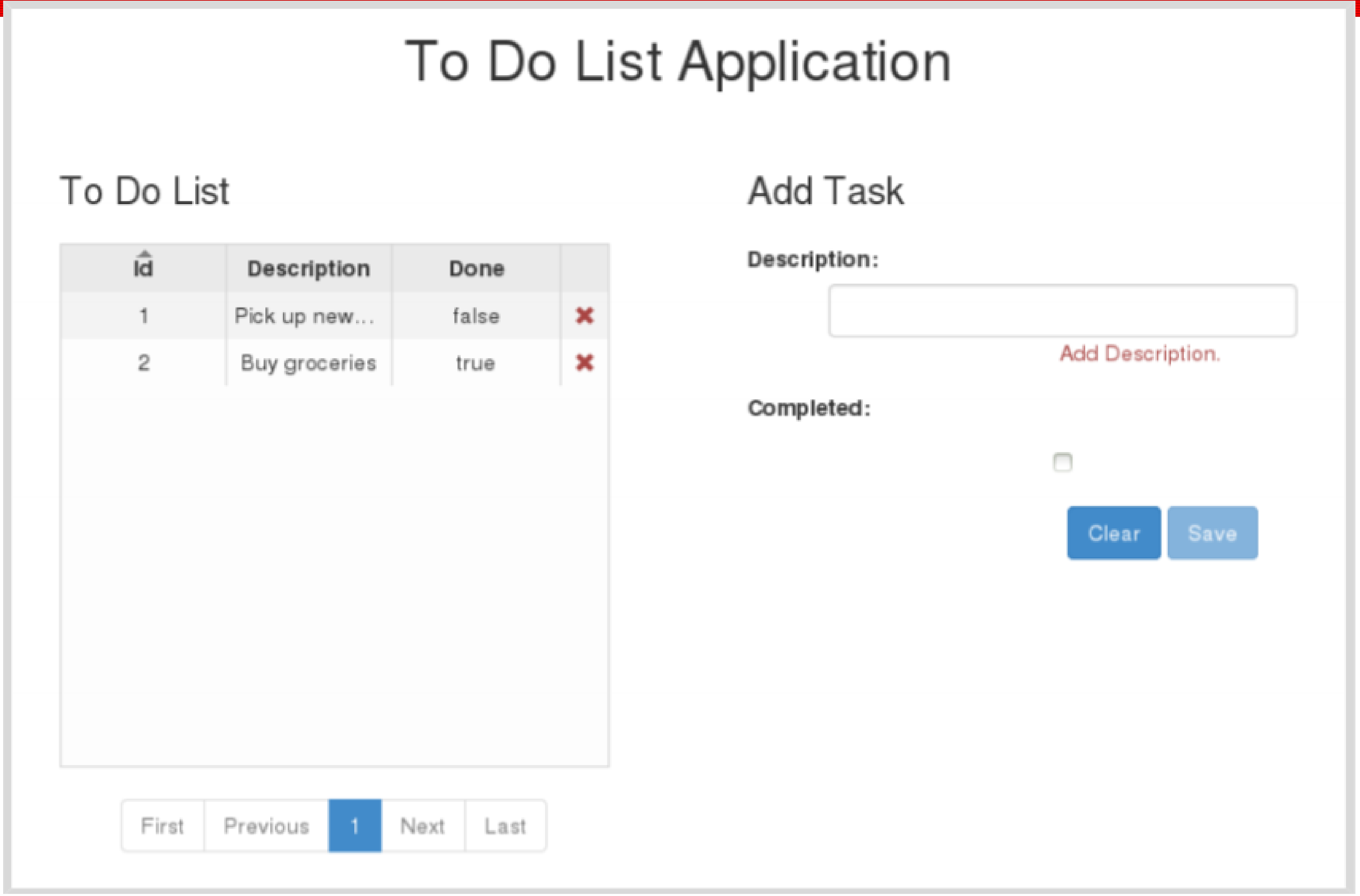
Multi-Container Applications for Rootless containers



Describing To Do List Application



The To Do List application



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 Containers

In this lab, students will

- Create a script that start a Node.js application container
- Create a script that start MySQL container
- Connect application container and MySQL container

Deploying a Multi-Container Application on OpenShift

After completing this section, you will be:

Examining the

- Skeleton of a Template

Processing

- a Template Using the CLI

Configuring

- Persistent Storage for OpenShift Applications

Examining the Skeleton of a Template

Deploying applications requires creating several resources

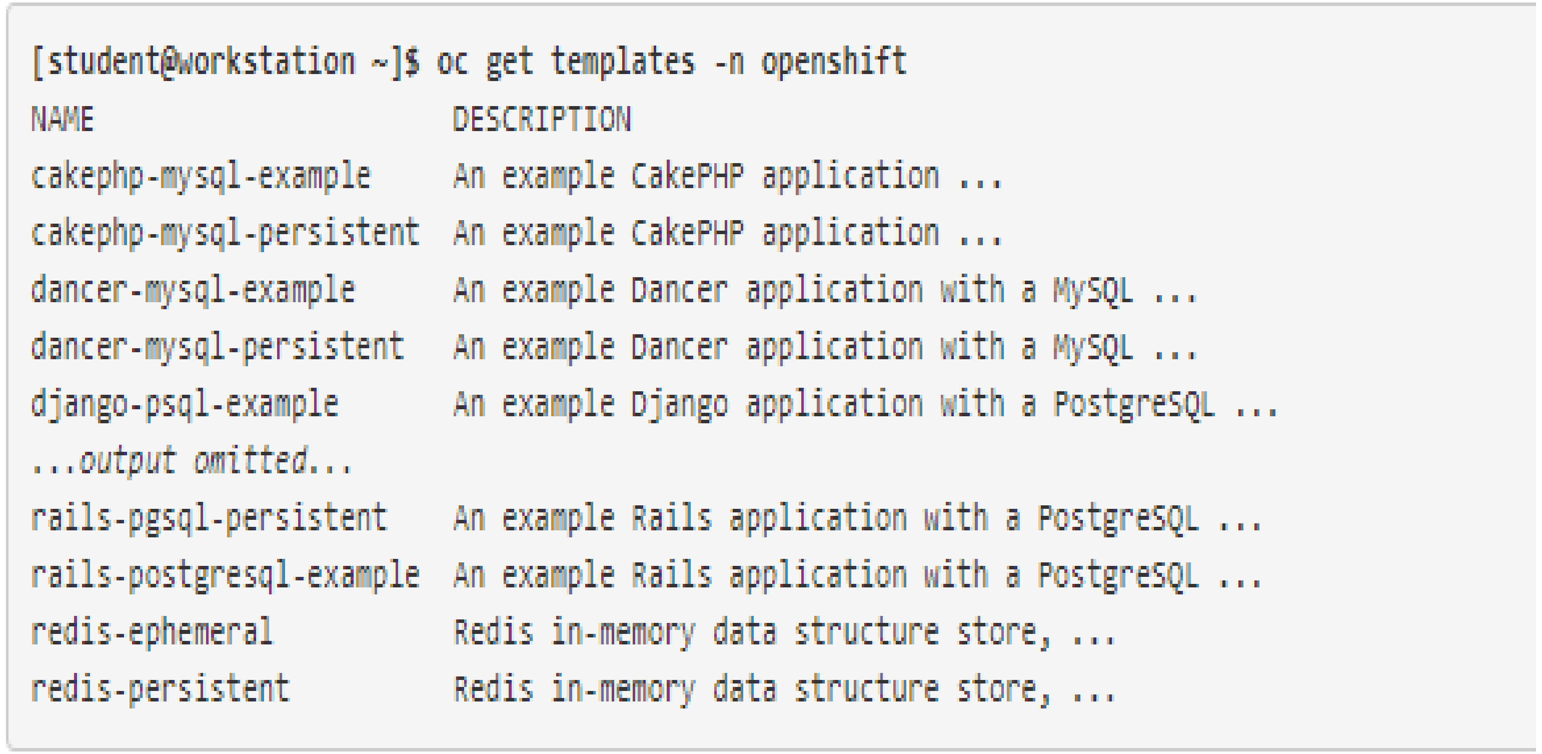
- BuildConfig
- Deployment
- DeploymentConfig
- Service
- Quotas
- Limits
- Routes
- Pods

Template

- simplify resources creation
- Reusable
- Can be processed by dynamic parameters

OpenShift templates

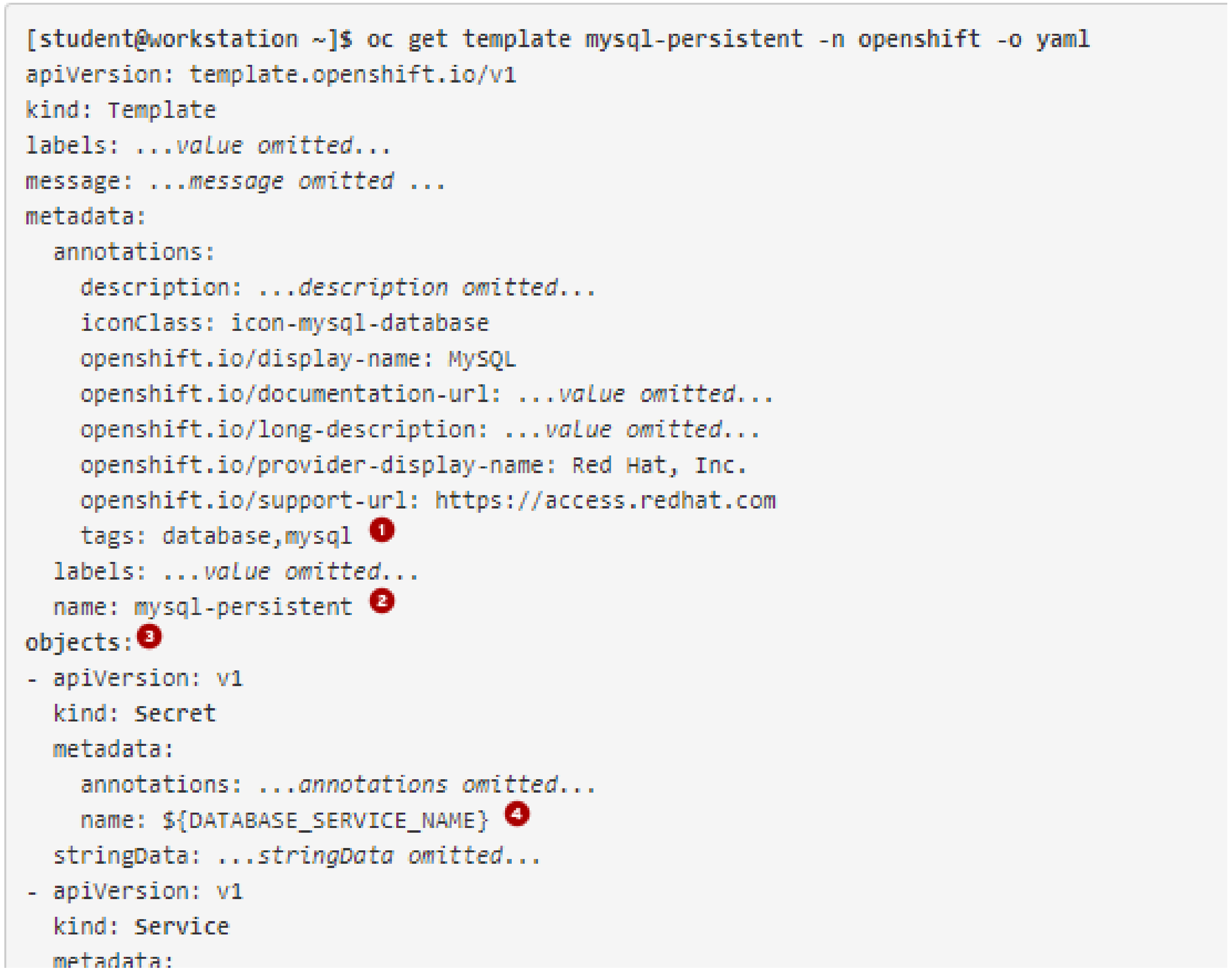
- List installed templates by oc get templates –n openshift project



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

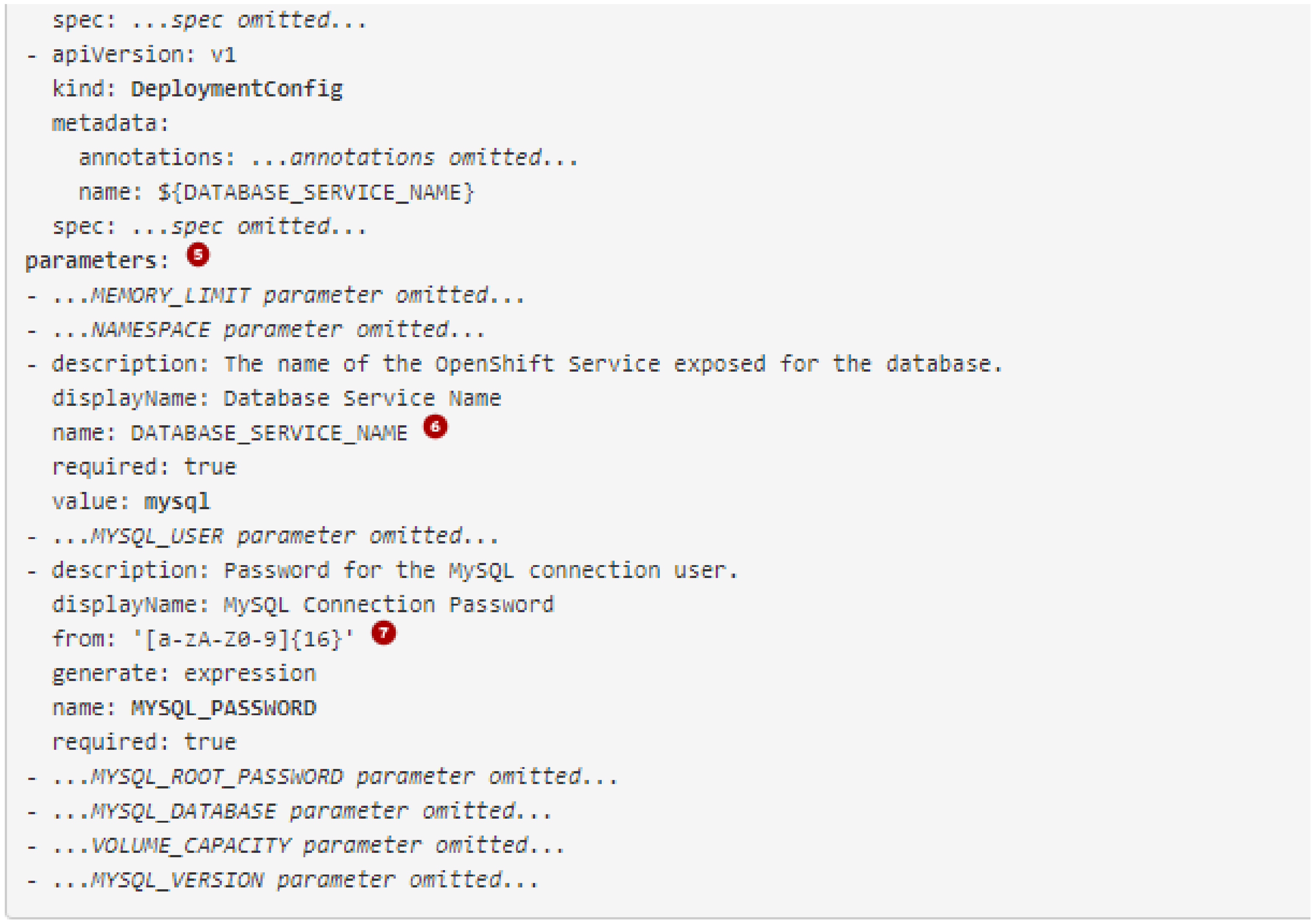
- Extract a particular yaml template definition

Extract a particular yaml template definition



```
[student@workstation ~]$ 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 ❶
  labels: ...value omitted...
  name: mysql-persistent ❷
objects:❸
- apiVersion: v1
  kind: Secret
  metadata:
    annotations: ...annotations omitted...
    name: ${DATABASE_SERVICE_NAME} ❹
  stringData: ...stringData omitted...
- apiVersion: v1
  kind: Service
  metadata:
```

Extract a particular yaml template definition



```
  spec: ...spec omitted...
- apiVersion: v1
  kind: DeploymentConfig
  metadata:
    annotations: ...annotations omitted...
    name: ${DATABASE_SERVICE_NAME}
  spec: ...spec omitted...
parameters: ❺
- ...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 ❻
  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}' ❼
  generate: expression
  name: MYSQL_PASSWORD
  required: true
- ...MYSQL_ROOT_PASSWORD parameter omitted...
- ...MYSQL_DATABASE parameter omitted...
- ...VOLUME_CAPACITY parameter omitted...
- ...MYSQL_VERSION parameter omitted...
```

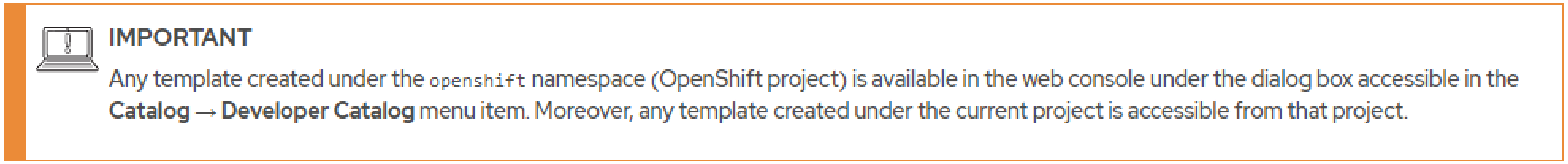
Publish your own template

- Enables re-deployment of your applications
- Use oc create -f command to publish in current project

```
$ oc create -f todo-template.yaml
```

- Publish to openshift project

```
$ oc create -f todo-template.yaml -n openshift
```



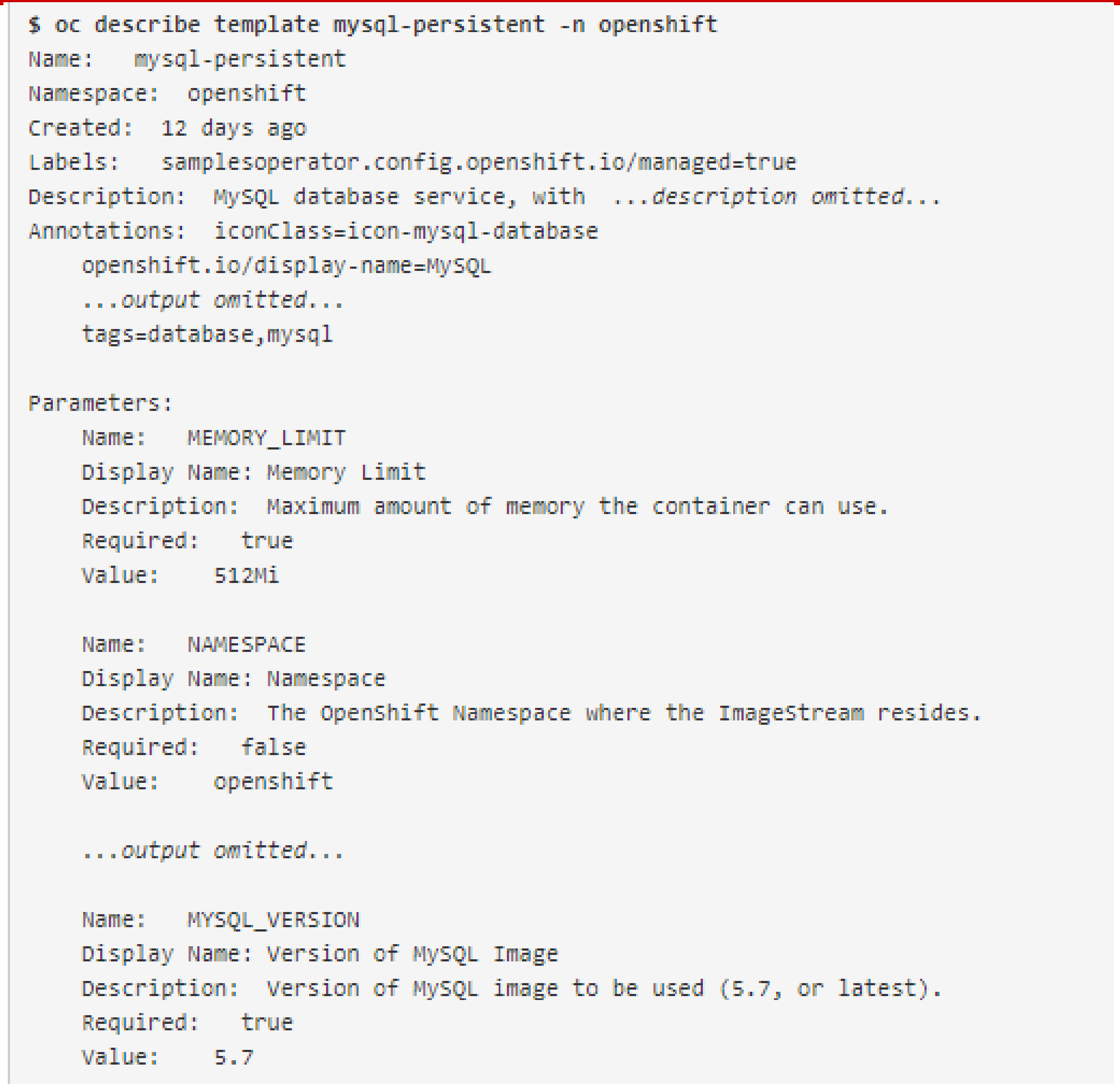
IMPORTANT

Any template created under the `openshift` namespace (OpenShift project) is available in the web console under the dialog box accessible in the **Catalog → Developer Catalog** menu item. Moreover, any template created under the current project is accessible from that project.

Parameters

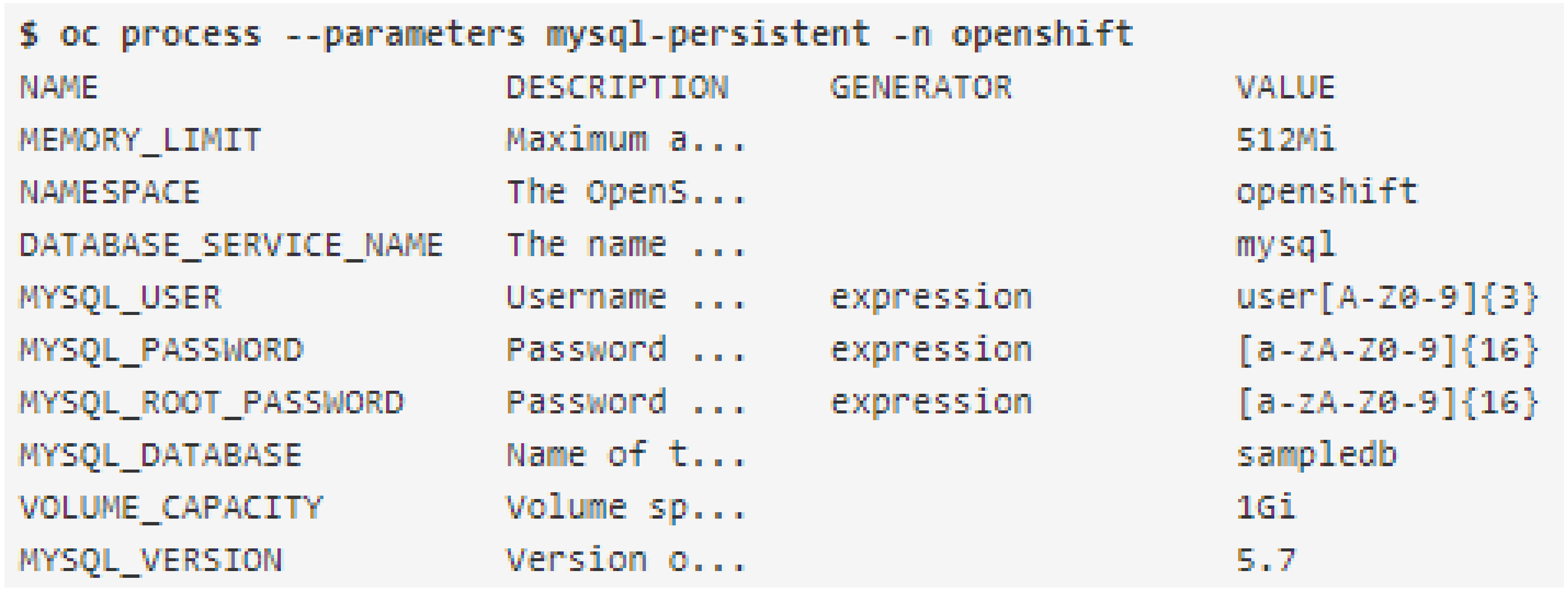
- Which requires user input
- Named parameters
- Default values
- Use oc process subcommand to process parameter configuration
- To list available parameters from template
    - ❑ Use oc describe subcommand
    - ❑ Use oc process --parameters subcommand

oc describe



```
$ 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
```

oc process --parameters



```
$ 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...                        5.7
```

Processing a Template Using the CLI

- Process a template file and generate output using JSON

```
$ oc process –f <filename>
```

- Process a template file and generate output using YAML instead

```
$ oc process –o yaml –f <filename>
```

- Process a template file from current project

```
$ oc process <published template name>
```

- Process a template file from openshift project

```
$ oc process <published template name> -n openshift
```

Process a template and deploy it

- Process following mysql.yaml template file by assigning values to configurable parameters . Instead of just output to terminal, write output into mysqlProcessed.yaml

```
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
```

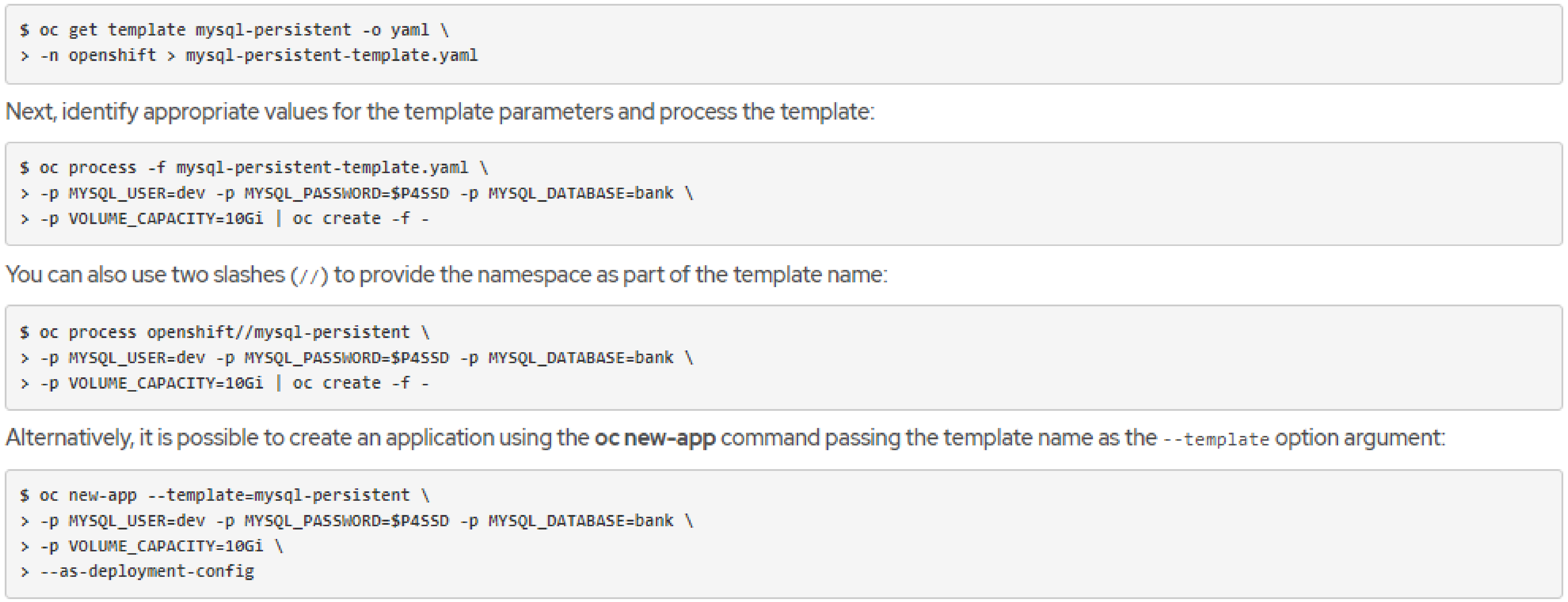
- Use oc create subcommand to create application from the mysqlProcessed.yaml file

```
oc create  –f mysqlProcessed.yaml
```

- Alternatively, directly create application while process the template

```
oc process -o yaml -f mysql.yaml \
    -p MYSQL_USER=dev -p MYSQL_PASSWORD=$P4SSD -p MYSQL_DATABASE=bank \
    -p VOLUME_CAPACITY=10Gi | oc create –f -
```

Process a template from openshift project



```
$ oc get template mysql-persistent -o yaml \
> -n openshift > mysql-persistent-template.yaml
```

Next, identify appropriate values for the template parameters and process the template:

```
$ 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 -
```

You can also use two slashes (`//`) to provide the namespace as part of the template name:

```
$ oc process openshift//mysql-persistent \
> -p MYSQL_USER=dev -p MYSQL_PASSWORD=$P4SSD -p MYSQL_DATABASE=bank \
> -p VOLUME_CAPACITY=10Gi | oc create -f -
```

Alternatively, it is possible to create an application using the **oc new-app** command passing the template name as the `--template` option argument:

```
$ oc new-app --template=mysql-persistent \
> -p MYSQL_USER=dev -p MYSQL_PASSWORD=$P4SSD -p MYSQL_DATABASE=bank \
> -p VOLUME_CAPACITY=10Gi \
> --as-deployment-config
```

Persistent Storage with PV, PVC, SC

Cluster Admin

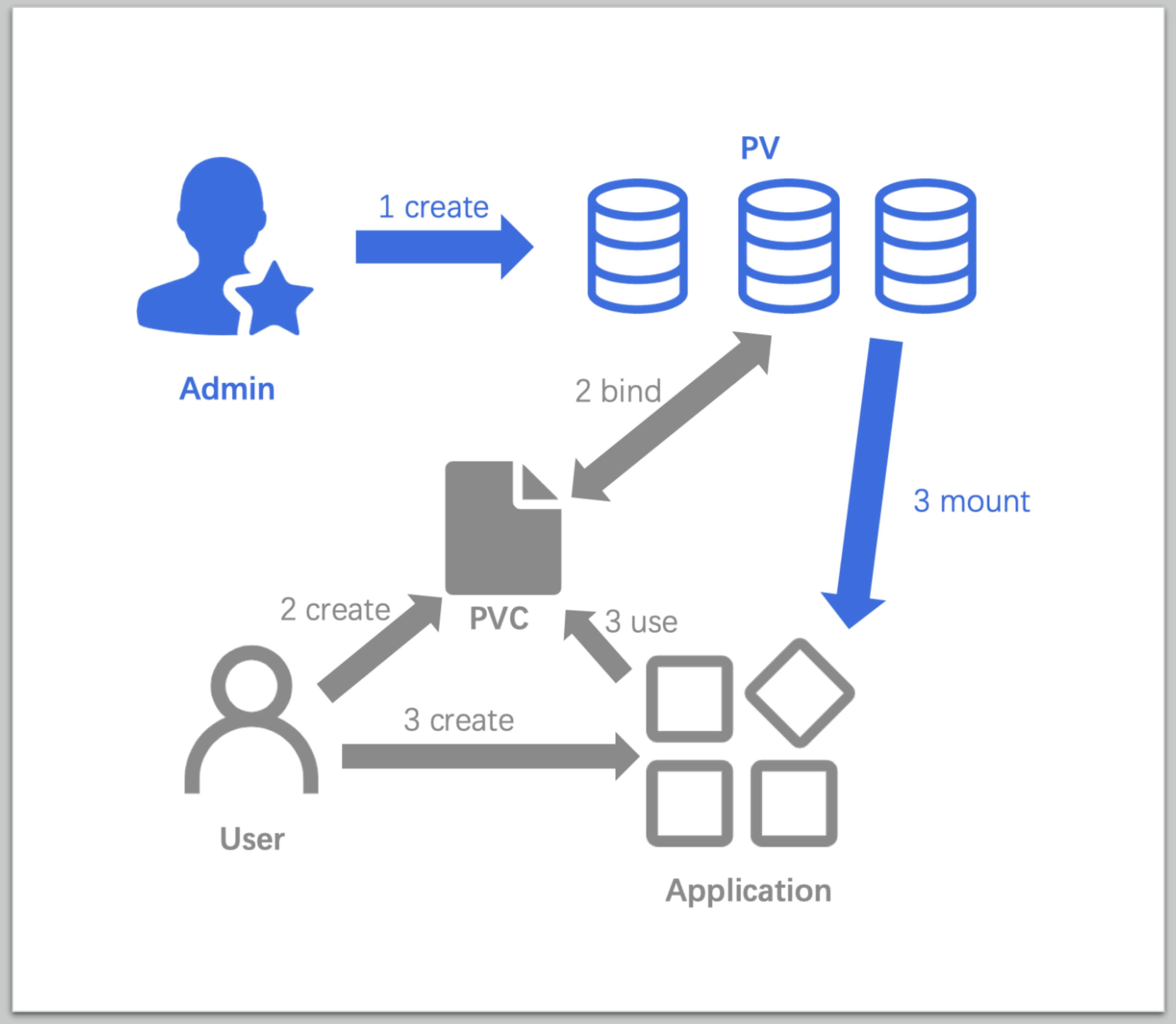
1. Create PV/SC connecting to actual physical storage

- NFS, GCE PD, Azure Block Storage, AWS EBS, AWS EFS, Netapp StorageGrod and so on

2. DevOps or User create PVC to bind application deployment to respective PV

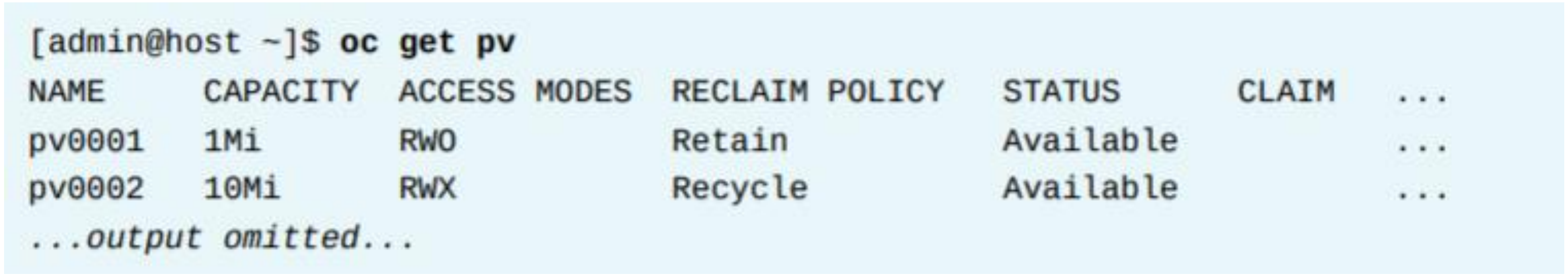
- Read / Write mode
- Size of storage area
- Other parameters

3. Once successful, application mount PV to folder and start writing



Configuring Persistent Storage for OpenShift Applications

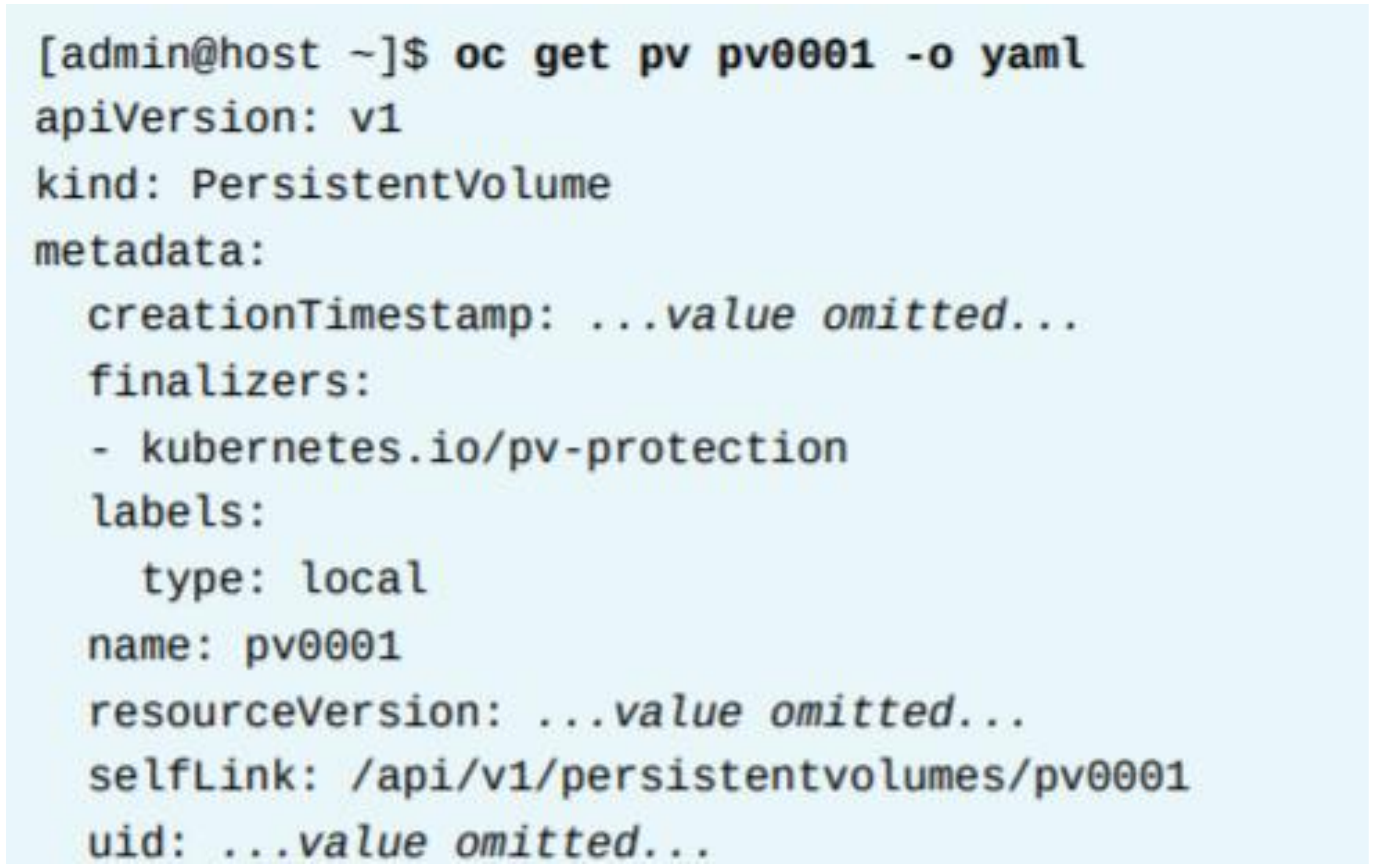
- Clustered wide resource
- Must use Cluster Admin role
- To list Persistent Storage



```
[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...
```

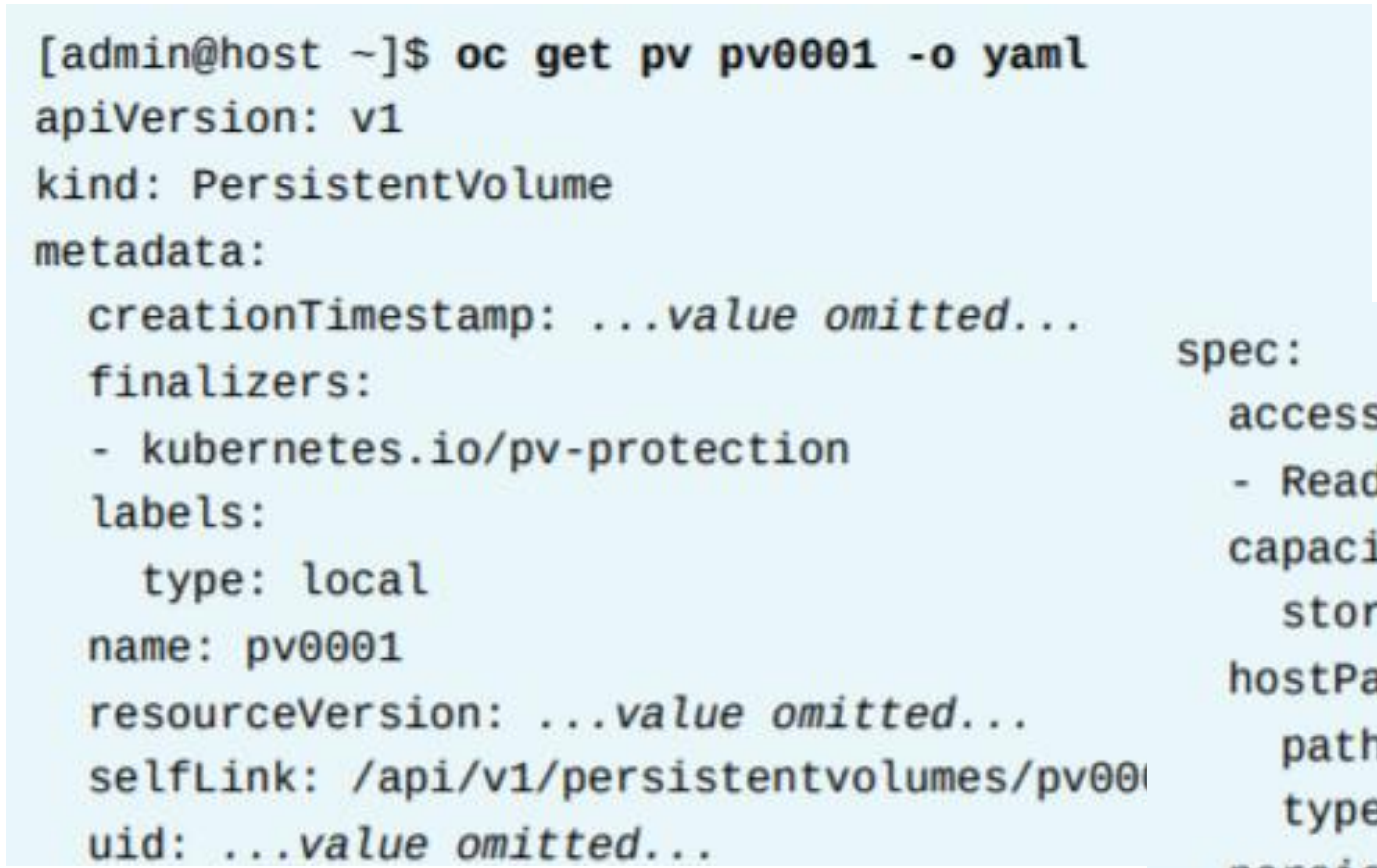
Extract PV yaml

- To see YAML definition for given PV

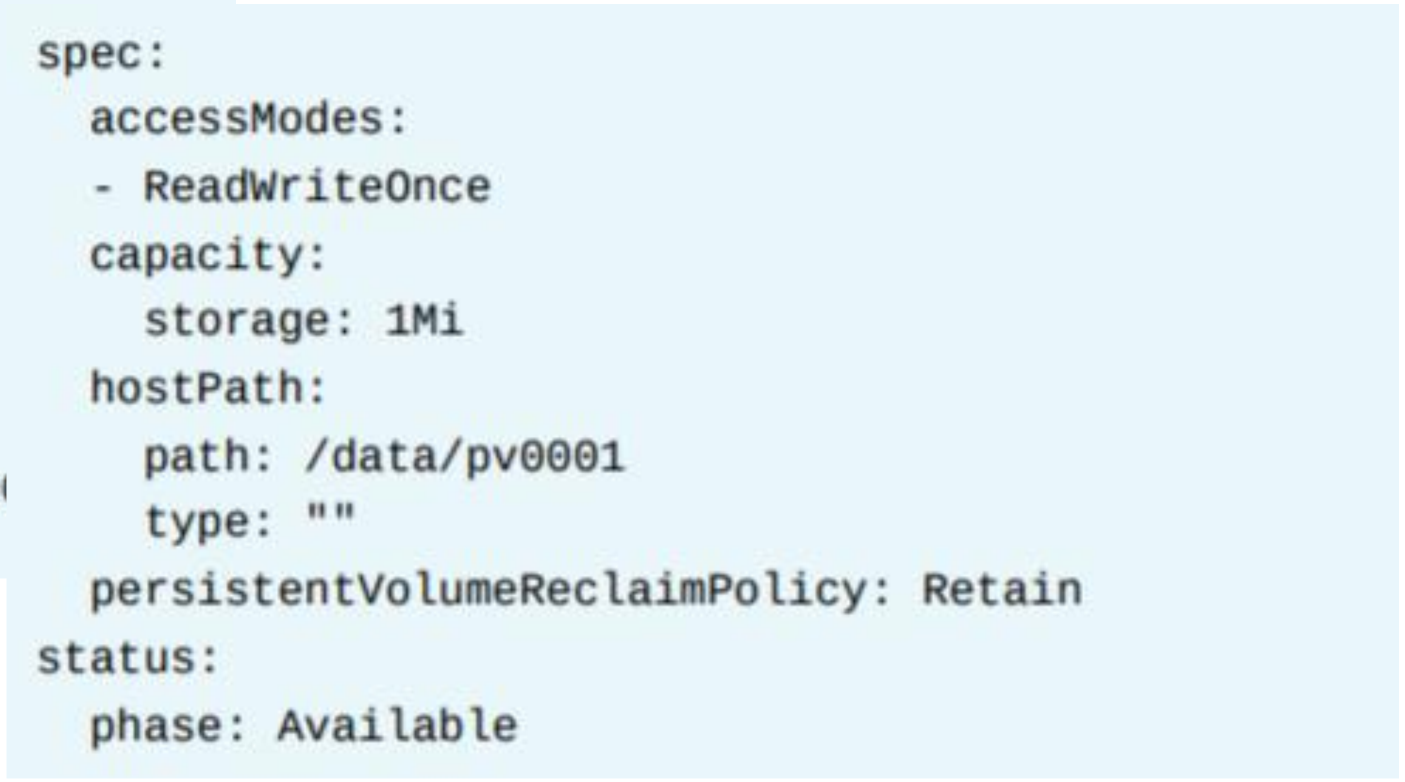


```
[admin@host ~]$ oc get pv pv0001 -o yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  creationTimestamp: ...value omitted...
  finalizers:
  - kubernetes.io/pv-protection
  labels:
    type: local
  name: pv0001
  resourceVersion: ...value omitted...
  selfLink: /api/v1/persistentvolumes/pv0001
  uid: ...value omitted...
```

Extract PV yaml

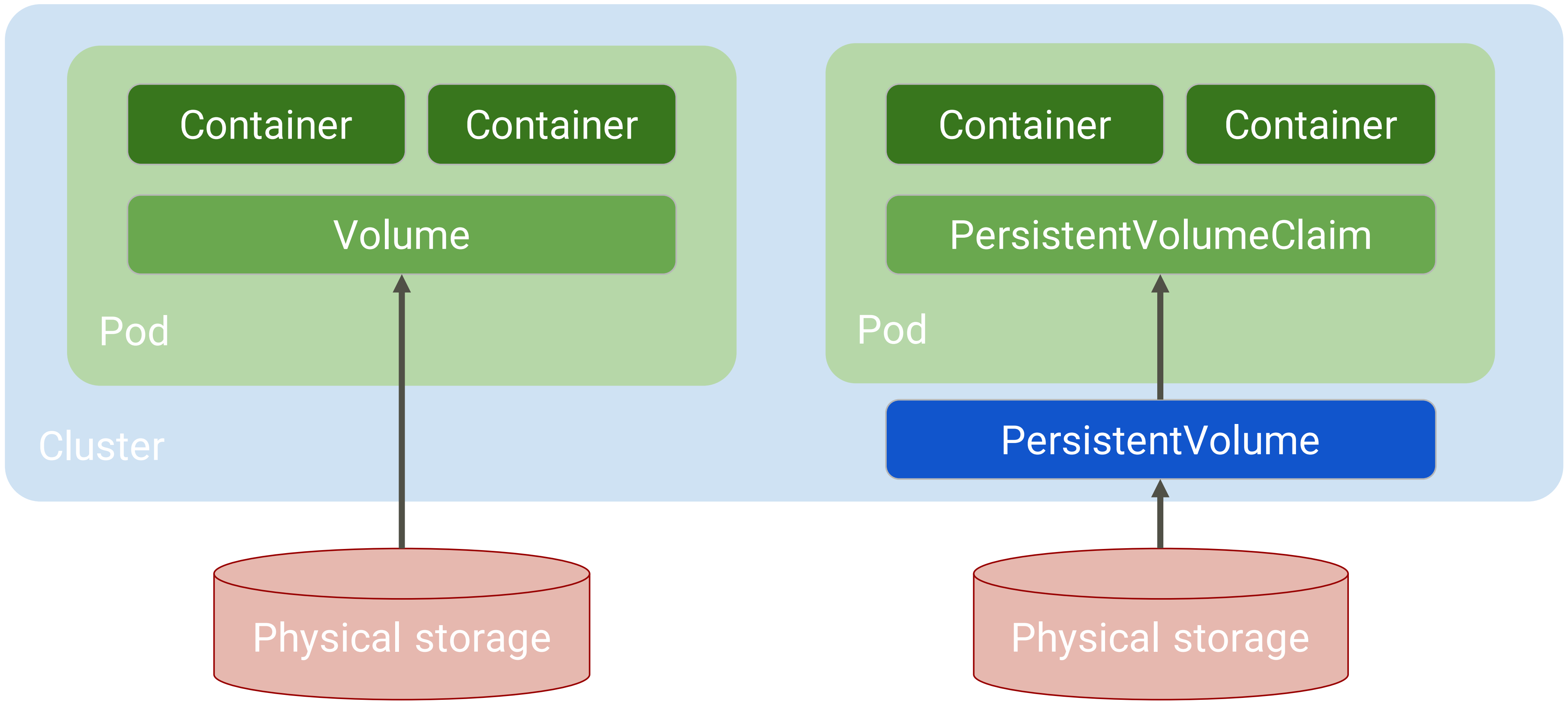


```
[admin@host ~]$ oc get pv pv0001 -o yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  creationTimestamp: ...value omitted...
  finalizers:
  - kubernetes.io/pv-protection
  labels:
    type: local
  name: pv0001
  resourceVersion: ...value omitted...
  selfLink: /api/v1/persistentvolumes/pv00
  uid: ...value omitted...
```



```
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 1Mi
  hostPath:
    path: /data/pv0001
    type: ""
  persistentVolumeReclaimPolicy: Retain
status:
  phase: Available
```

Requesting Persistent Volumes



Creating a PersistentVolume manifest

- Defined by the Storage Admin / Cluster Admin

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pd-volume
spec:
        storageClassName: "standard"
        capacity:
          storage: 100G
        accessModes:
        - ReadWriteOnce:
        gcePersistentDisk:
          pdName: demo-disk
          fsType: ext4
```

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: standard
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard
  replication-type: none
```

PVC StorageClassName
must match the
PV StorageClassName

Requesting Persistent Volumes

- Developer / DevOps create PVC object to request a dedicated storage resource from cluster pool.

```
apiVersion: v1
kind: Pod
metadata:
   name: demo-pod
spec:
   containers:
   - name: demo-container
        image: gcr.io/hello-app:1.0
        volumeMounts:
        - mountPath: /demo-pod
        name: pd-volume
   volumes:
   - name: pd-volume
     PersistentVolumeClaim:
     claimName: pd-volume-claim
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
   name: pd-volume-claim
spec:
   storageClassName: “standard"
   accessModes:
   - ReadWriteOnce:
   resources:
        requests:
        storage: 100G
```

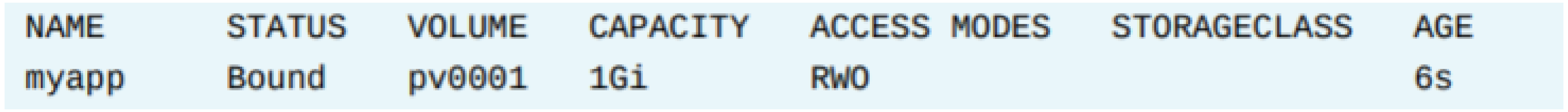
Create and list PVC

- Defines storage requirements for application such as capacity or throughput

```
$ oc create –f pvc.yaml
```

- OpenShift attempt to find available PersistentVolume that able satisfy above PVC requirements. If there is a match, it binds PersistentVolume object to the PVC
- To list PVCs in project

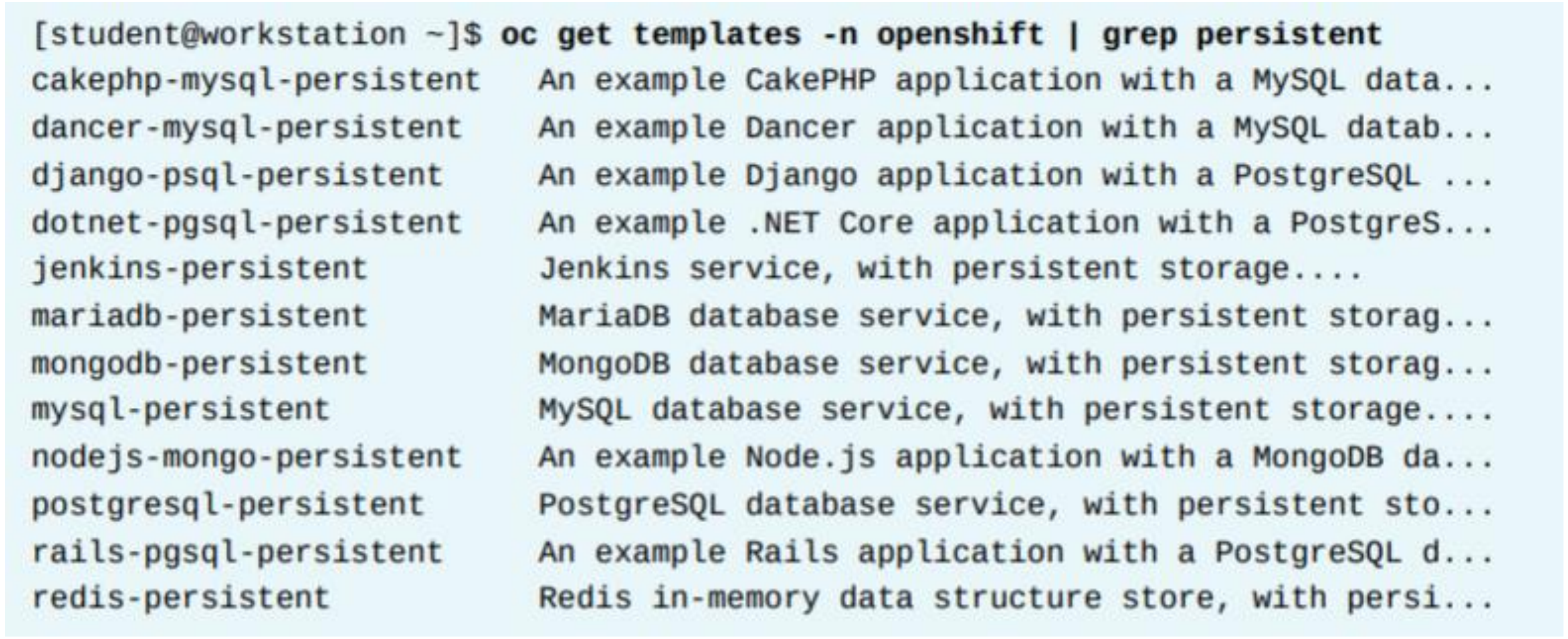
```
$ oc get pvc
```



```
NAME      STATUS    VOLUME    CAPACITY   ACCESS MODES   STORAGECLASS   AGE
myapp     Bound     pv0001    1Gi        RWO                           6s
```

Configure Persistent Storage with Templates

- Templates used to simplify creation of applications requiring persistent storage
- Suffix of **–persistent**

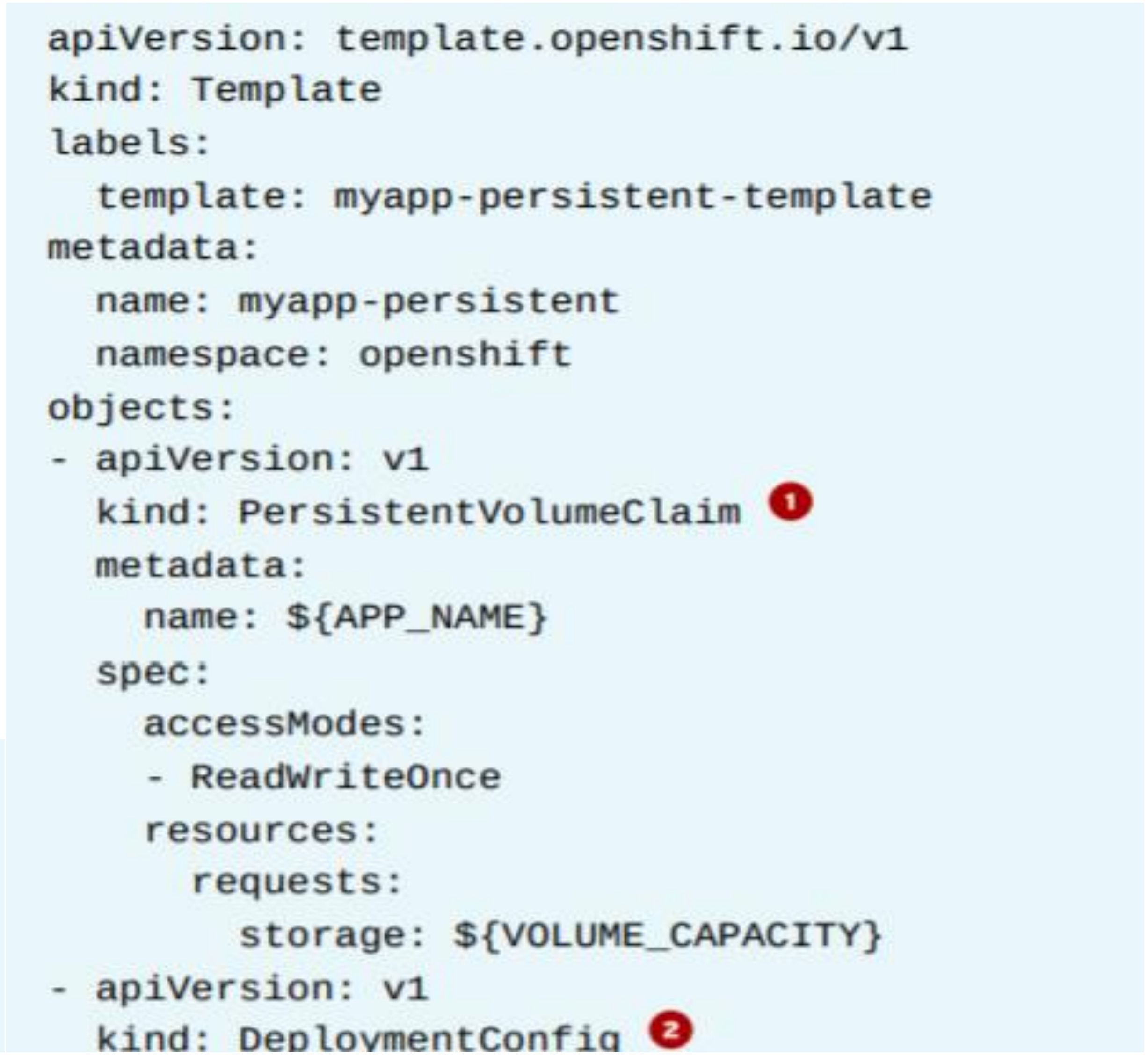


```
[student@workstation ~]$ oc get templates -n openshift | grep persistent
cakephp-mysql-persistent   An example CakePHP application with a MySQL data...
dancer-mysql-persistent    An example Dancer application with a MySQL datab...
django-psql-persistent     An example Django application with a PostgreSQL ...
dotnet-pgsql-persistent    An example .NET Core application with a PostgreS...
jenkins-persistent         Jenkins service, with persistent storage....
mariadb-persistent         MariaDB database service, with persistent storag...
mongodb-persistent         MongoDB database service, with persistent storag...
mysql-persistent           MySQL database service, with persistent storage....
nodejs-mongo-persistent    An example Node.js application with a MongoDB da...
postgresql-persistent      PostgreSQL database service, with persistent sto...
rails-pgsql-persistent     An example Rails application with a PostgreSQL d...
redis-persistent           Redis in-memory data structure store, with persi...
```

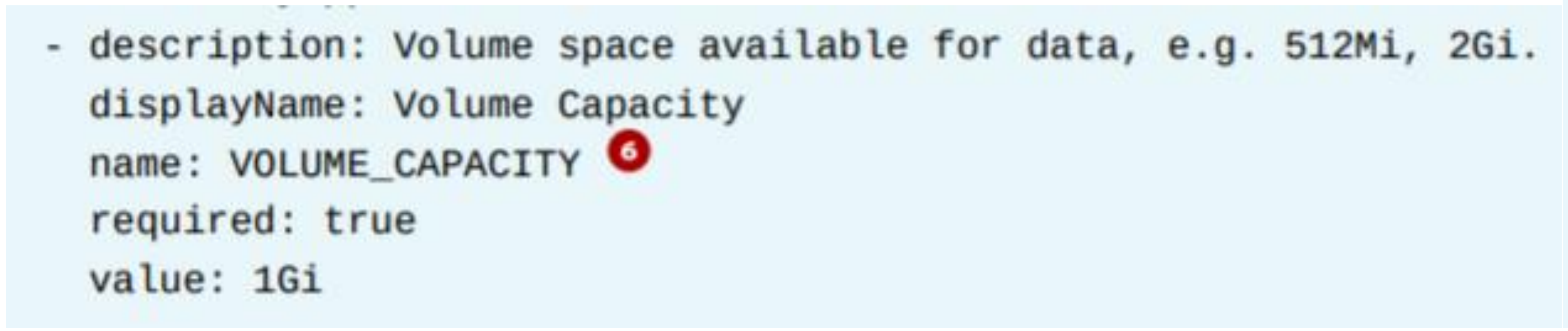
Example: template with persistent storage

```
$ oc describe template -n openshift cakephp-mysql-persistent | grep -i volume -A6
```

Name:	VOLUME_CAPACITY
Display Name:	Volume Capacity
Description:	Volume space available for data, …
Required:	true
Value:	1Gi



```
apiVersion: template.openshift.io/v1
kind: Template
labels:
  template: myapp-persistent-template
metadata:
  name: myapp-persistent
  namespace: openshift
objects:
- apiVersion: v1
  kind: PersistentVolumeClaim ❶
  metadata:
    name: ${APP_NAME}
  spec:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: ${VOLUME_CAPACITY}
- apiVersion: v1
  kind: DeploymentConfig ❷
```



```
- description: Volume space available for data, e.g. 512Mi, 2Gi.
  displayName: Volume Capacity
  name: VOLUME_CAPACITY ❻
  required: true
  value: 1Gi
```

Guided Exercise: Creating an Application with a Template

In this exercise, you will

- Deploy the To do List application in OCP using template

Lab: Deploy Multi-Container Applications

You should be able to:

- Deploy a PHP Application with a MySQL database using OpenShift template
- Define resources needed by the application