Guided Exercise: Provision Persistent Data Volumes

Deploy a MySQL database with persistent storage from a PVC and identify the PV and storage provisioner that back the application.

Outcomes

- Deploy a MySQL database with persistent storage from a PVC.
- Identify the PV that backs the application.
- Identify the storage provisioner that created the PV.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise. This command ensures that the cluster is accessible and that all resources are available for this exercise.

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

Instructions

1. Log in to the OpenShift web console as the `developer` user and switch to the **Administrator** perspective.

   Open a web browser and go to `https://console-openshift-console.apps.ocp4.example.com`

   Click **Red Hat Identity Management** and log in as the `developer` user with `developer` as the password. If the **Welcome to the Developer Perspective** message is displayed, then click **Skip tour**.

   From the perspective switcher, select **Administrator**.

2. Identify the default storage class by going to **Storage** → **StorageClasses**. The `nfs-storage` storage class has the `Default` label and uses the `k8s-sigs.io/nfs-subdir-external-provisioner` provisioner.

3. Use the `registry.ocp4.example.com:8443/rhel8/mysql-80` container image to create a MySQL deployment named `db-pod`. Use the `storage-volumes` project that the `lab` command created. Add a service for the database.

   Go to **Workloads** → **Deployments**.

   Set the project to `storage-volumes` and click **Create Deployment**.

   Use `db-pod` for the deployment name and change the image name to `registry.ocp4.example.com:8443/rhel8/mysql-80`.

   Add the required environment variables for the database.

   Table 5.5. MySQL Environment Variables

Name	Value
`MYSQL_USER`	`user1`
`MYSQL_PASSWORD`	`redhat123`
`MYSQL_DATABASE`	`items`

   In the **Advanced options** section, click **Scaling** and set the number of replicas to `1`.

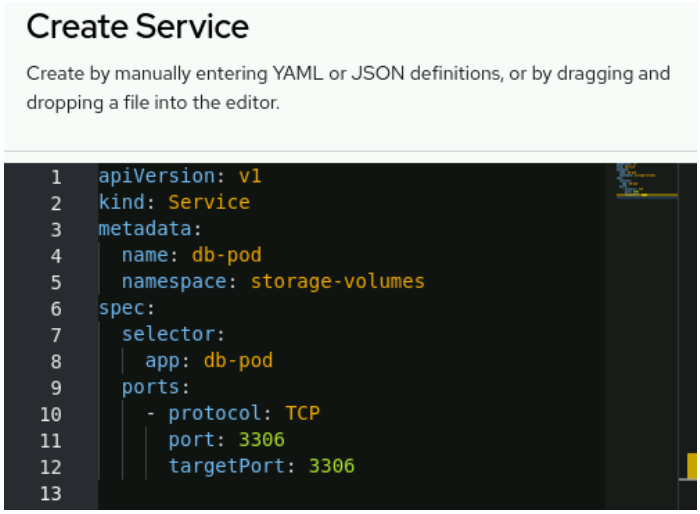
   Click **Create**. Wait for the blue circle to indicate that a single pod is running.

Create a service for the database. Go to **Networking** → **Services** and click **Create Service**.

Update the resource file with the following values:

Table 5.6. Service Fields

Field name	Value
Name	`db-pod`
Selector	`app=db-pod`
Port	`3306`
Target port	`3306`



Click **Create**.

4. Add to the deployment a 1 GiB, `ReadWriteOnce` (`RWO`) PVC named `db-pod-pvc`. Set the `/var/lib/mysql` directory as the mount path.

   Go to **Workloads** → **Deployments**.

   Click the vertical ellipsis icon (⋮) in the row for the `db-pod` deployment, and select **Add storage**.

   In the `Add Storage` form, click **Create new claim**.

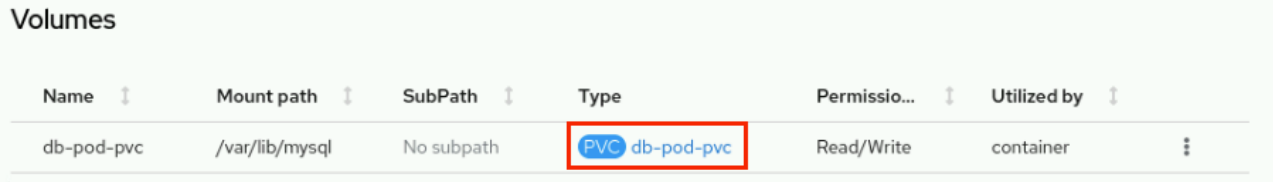
   Complete the form with the following values:

   Table 5.7. Add PVC Storage Fields

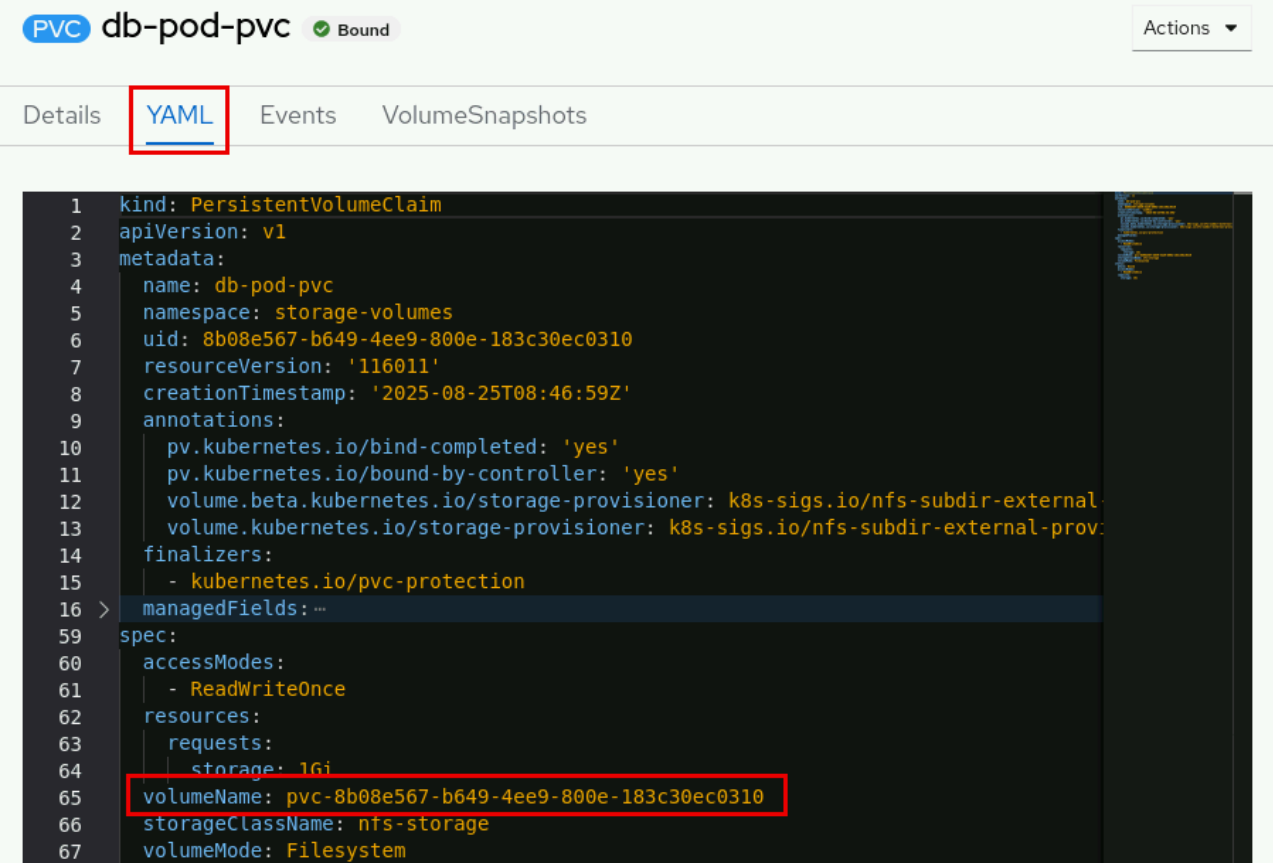
Field name	Value
Persistent volume claim name	`db-pod-pvc`
Access mode	`RWO`
Size	1 GiB
Volume mode	`Filesystem`
Mount path	`/var/lib/mysql`

   Click **Save**.

   On the `Deployment details` page, scroll to the `Volumes` section, and click the `db-pod-pvc` link to view the PVC details.



To retrieve the name of the PV that is associated with the PVC, go to the **YAML** tab. The name of the PV is available in the `.spec.volumeName` parameter. The name starts with `pvc-` even though the resource is a PV.



The name of the PV is different on your system.

5. Verify that configuring storage changes the `deployment` resource.

   Go to **Workloads** → **Deployments**, click the `db-pod` deployment name, and go to the **YAML** tab.

   Verify that configuring storage for the deployment results in the `volumes` and `volumeMounts` additions.

```
kind: Deployment
apiVersion: apps/v1

...output omitted...

      volumes:
        - name: db-pod-pvc
          persistentVolumeClaim:
            claimName: db-pod-pvc

...output omitted...

        volumeMounts:
          - name: db-pod-pvc
            mountPath: /var/lib/mysql

...output omitted...
```

6. Use a configuration map resource to add initialization data to the database.

   Review the `~/DO180/labs/storage-volumes/configmap/init-db.sql` script that initializes the database. You do not have to change its contents.

```
DROP TABLE IF EXISTS `Item`;
CREATE TABLE `Item` (`id` BIGINT not null auto_increment primary key, `description` VARCHAR(100), `done` BIT);
INSERT INTO `Item` (`id`,`description`,`done`) VALUES (1,'Pick up newspaper', 0);
INSERT INTO `Item` (`id`,`description`,`done`) VALUES (2,'Buy groceries', 1);
```

In a terminal window, log in to the RHOCP cluster as the `developer` user with `developer` as the password.

```
[student@workstation ~]$ oc login -u developer -p developer \
  https://api.ocp4.example.com:6443
Login successful.
...output omitted...
```

Set the `storage-volumes` project as the active project.

```
[student@workstation ~]$ oc project storage-volumes
...output omitted...
```

Use the contents of the `init-db.sql` file to create a configuration map named `init-db-cm`.

```
[student@workstation ~]$ oc create configmap init-db-cm \
  --from-file \
  ~/DO180/labs/storage-volumes/configmap/init-db.sql
configmap/init-db-cm created
```

Add the `init-db-cm` configuration map as a volume named `init-db-volume` to the deployment. Specify the volume type as `configmap`, and set the `/var/db/config` directory as the mount path.

```
[student@workstation ~]$ oc set volumes deployment/db-pod \
  --add --name init-db-volume --type configmap \
  --configmap-name init-db-cm --mount-path /var/db/config
deployment.apps/db-pod volume updated
```

Start a remote shell session inside the container.

```
[student@workstation ~]$ oc rsh deployment/db-pod
sh-4.4$
```

Use the `mysql` client to execute the database script in the `/var/db/config/init-db` volume.

```
sh-4.4$ mysql -uuser1 -predhat123 \
  items < /var/db/config/init-db.sql
mysql: [Warning] Using a password on the command line interface can be insecure.
sh-4.4$
```

Execute a query to verify the database contents.

```
sh-4.4$ mysql -uuser1 -predhat123 items \
  -e 'select * from Item;'
mysql: [Warning] Using a password on the command line interface can be insecure.
+----+------------------+------------+
| id | description      | done       |
+----+------------------+------------+
|  1 | Pick up newspaper | 0x00       |
|  2 | Buy groceries    | 0x01       |
+----+------------------+------------+
sh-4.4$
```

Exit the shell session.

```
sh-4.4$ exit
exit
```

7. Delete and then re-create the `db-pod` deployment to verify that the database data persists.

   Delete the `db-pod` deployment.

   ```
   [student@workstation ~]$ oc delete deployment/db-pod
   deployment.apps "db-pod" deleted
   ```

   Verify that the PVC still exists without the deployment.

```
[student@workstation ~]$ oc get pvc
NAME         STATUS   VOLUME            CAPACITY   ...
db-pod-pvc   Bound    pvc-8b08...0310   1Gi        ...
```

Re-create the `db-pod` deployment.

```
[student@workstation ~]$ oc create deployment db-pod \
  --port 3306 \
  --image registry.ocp4.example.com:8443/rhel8/mysql-80
deployment.apps/db-pod created
```

Add the environment variables.

```
[student@workstation ~]$ oc set env deployment/db-pod \
  MYSQL_USER=user1 \
  MYSQL_PASSWORD=redhat123 \
  MYSQL_DATABASE=items
deployment.apps/db-pod updated
```

Use the `oc set volumes` command to attach the existing PVC to the deployment.

```
[student@workstation ~]$ oc set volumes deployment/db-pod \
  --add --type pvc \
  --mount-path /var/lib/mysql \
  --name db-pod-vol \
  --claim-name db-pod-pvc
deployment.apps/db-pod volume updated
```

Create a `query-db` pod by using the `oc run` command and the `registry.ocp4.example.com:8443/redhattraining/do180-dbinit` container image. Use the pod to execute a query against the database service.

It takes up to one minute for the command to display the result.

```
[student@workstation ~]$ oc run query-db -it --rm --image \
  registry.ocp4.example.com:8443/redhattraining/do180-dbinit \
  --restart Never \
  -- /bin/bash -c "mysql -uuser1 -predhat123 --protocol tcp \
  -h db-pod -P3306 items -e 'select * from Item;'"
mysql: [Warning] Using a password on the command line interface can be insecure.
+----+-------------------+------------+
| id | description       | done       |
+----+-------------------+------------+
|  1 | Pick up newspaper | 0x00       |
|  2 | Buy groceries     | 0x01       |
+----+-------------------+------------+
pod "query-db" deleted
```

8. Delete the `db-pod` deployment and the `db-pod-pvc` PVC.

   Delete the `db-pod` deployment.

   ```
   [student@workstation ~]$ oc delete deployment/db-pod
   deployment.apps "db-pod" deleted
   ```

   Delete the `db-pod-pvc` PVC.

   ```
   [student@workstation ~]$ oc delete pvc/db-pod-pvc
   persistentvolumeclaim "db-pod-pvc" deleted
   ```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

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