Guided Exercise: Manage Non-shared Storage with Stateful Sets

Deploy a replicated web server by using a deployment and verify that all web server pods share a PV; and deploy a replicated MySQL database by using a stateful set and verify that each database instance gets a dedicated PV.

Outcomes

- Deploy a web server with persistent storage.
- Add data to the persistent storage.
- Scale the web server deployment and observe the data that is shared with the replicas.
- Create a database server with a headless service and a stateful set by using YAML manifest files.
- Verify that each instance from the stateful set has a persistent volume claim.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that all resources are available for this exercise.

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

Instructions

1. Create a web server deployment named `web-server`. Use the `registry.ocp4.example.com:8443/redhattraining/hello-world-nginx:latest` container image.

   Log in to the OpenShift cluster as the `developer` user with `developer` as the password.

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

   Change to the `storage-statefulsets` project.

   ```
   [student@workstation]$ oc project storage-statefulsets
   Now using project "storage-statefulsets" on server ...output omitted...
   ```

   Create the `web-server` deployment.

   ```
   [student@workstation ~]$ oc create deployment web-server \
     --image registry.ocp4.example.com:8443/redhattraining/hello-world-nginx:latest
   deployment.apps/web-server created
   ```

   Verify the deployment status.

   ```
   [student@workstation ~]$ oc get deployment
   NAME         READY   UP-TO-DATE   AVAILABLE   AGE
   web-server   1/1     1            1           50s
   ```

2. Add the PVC named `web-pv` to the `web-server` deployment. Use the default storage class and the following information to create the persistent volume claim:

Field	Value
Name	`web-pv`
Type	`persistentVolumeClaim`
Claim mode	`rwo`
Claim size	`5Gi`
Mount path	`/usr/share/nginx/html`
Claim name	`web-pv-claim`

   Use the `oc set volumes` command to create and add the `web-pv` persistent volume to the `web-server` deployment.

```
[student@workstation ~]$ oc set volumes deployment/web-server \
  --add --name web-pv --type persistentVolumeClaim --claim-mode rwo \
  --claim-size 5Gi --mount-path /usr/share/nginx/html --claim-name web-pv-claim
deployment.apps/web-server volume updated
```

Because a storage class was not specified with the `--claim-class` option, the command uses the default storage class to create the persistent volume.

Verify the deployment pod status. Notice that a new pod is created and in running status.

```
[student@workstation ~]$ oc get pods -l app=web-server
NAME                          READY   STATUS    RESTARTS  AGE
web-server-5fb94fd568-48n8t   1/1     Running   0         5s
```

Check that the `web-server` pod mounts the `web-pv` PVC. The pod name is different in your cluster.

```
[student@workstation ~]$ oc get pod web-server-5fb94fd568-48n8t -o json | \
  jq .spec.volumes[0].persistentVolumeClaim.claimName
"web-pv-claim"
```

Verify that the `web-pv` PVC shows the bound status.

```
[student@workstation ~]$ oc get pvc
NAME           STATUS   VOLUME     CAPACITY  ACCESS MODES  STORAGECLASS ...
web-pv-claim   Bound    pvc-42...  5Gi       RWO           nfs-storage  ...
```

The default storage class, `nfs-storage`, provisioned the persistent volume.

3. Add data to the `web-pv` PVC by using the `exec` command.

   List the pods to retrieve the `web-server` pod name.

   ```
   [student@workstation ~]$ oc get pods
   NAME                          READY   STATUS    RESTARTS   AGE
   web-server-5fb94fd568-48n8t   1/1     Running   0          3m
   ```

   The pod name is different in your cluster.

   Use the `exec` command to add the pod name that you retrieved from the previous step to the `/usr/share/nginx/html/index.html` file on the pod.

   ```
   [student@workstation ~]$ oc exec -it pod/web-server-5fb94fd568-48n8t \
     -- /bin/bash -c \
     'echo "Hello, World from ${HOSTNAME}" > /usr/share/nginx/html/index.html'
   ```

   Retrieve the contents of the `/var/www/html/index.html` file to confirm that the pod name is in the file.

   ```
   [student@workstation ~]$ oc exec -it pod/web-server-5fb94fd568-48n8t \
     -- cat /usr/share/nginx/html/index.html
   Hello, World from web-server-5fb94fd568-48n8t
   ```

4. Scale the `web-server` deployment to two replicas and confirm that an additional pod is created.

   Scale the `web-server` deployment to two replicas.

   ```
   [student@workstation ~]$ oc scale deployment web-server --replicas 2
   deployment.apps/web-server scaled
   ```

   Verify the replica status and retrieve the pod names.

   ```
   [student@workstation ~]$ oc get pods
   NAME                          READY   STATUS    RESTARTS   AGE
   web-server-5fb94fd568-48n8t   1/1     Running   0          10m
   web-server-5fb94fd568-rqvpb   1/1     Running   0          6s
   ```

   The pod names are different in your cluster.

5. Retrieve the content of the `/usr/share/nginx/html/index.html` file on the `web-server` pods by using the `oc exec` command to verify that the file is the same in both pods.

Verify that the `/usr/share/nginx/html/index.html` file is the same in both pods.

```
[student@workstation ~]$ oc exec -it pod/web-server-5fb94fd568-48n8t \
  -- cat /usr/share/nginx/html/index.html
Hello, World from web-server-5fb94fd568-48n8t
```

```
[student@workstation ~]$ oc exec -it pod/web-server-5fb94fd568-rqvpb \
  -- cat /usr/share/nginx/html/index.html
Hello, World from web-server-5fb94fd568-48n8t
```

Notice that both files show the name of the first instance, because they share the persistent volume.

6. Create a database server by using a stateful set and a headless service.

   Edit the `/home/student/DO180/labs/storage-statefulsets/statefulset-db.yml` file and modify the placeholder text to match the following content:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: dbserver
spec:
  selector:
    matchLabels:
      app: database
  replicas: 2
  serviceName: mysql-svc
  template:
    metadata:
      labels:
        app: database
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: dbserver
        image: registry.ocp4.example.com:8443/redhattraining/mysql-app:v1
        ports:
        - name: database
          containerPort: 3306
        env:
        - name: MYSQL_USER
          value: "redhat"
        - name: MYSQL_PASSWORD
          value: "redhat123"
        - name: MYSQL_DATABASE
          value: "sakila"
        volumeMounts:
        - name: data
          mountPath: /var/lib/mysql
  volumeClaimTemplates:
  - metadata:
      name: data
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "lvms-vg1"
      resources:
        requests:
          storage: 1Gi
```

   Edit the `/home/student/DO180/labs/storage-statefulsets/service-db.yml` file and modify the placeholder text to match the following content:

```
apiVersion: v1
kind: Service
metadata:
  name: mysql-svc
  labels:
    app: database
spec:
  ports:
  - port: 3306
    name: mysql
  clusterIP: None
  selector:
    app: database
```

Create the `mysql-svc` headless service.

```
[student@workstation ~]$ oc create -f \
  /home/student/DO180/labs/storage-statefulsets/service-db.yml
service/mysql-svc created
```

Create the `dbserver` stateful set.

```
[student@workstation ~]$ oc create -f \
  /home/student/DO180/labs/storage-statefulsets/statefulset-db.yml
statefulset.apps/bdserver created
```

Run the `watch` command and wait until the two stateful set pods are running.

```
[student@workstation ~]$ watch oc get pod \
  -l app=database -o wide
NAME         READY   STATUS    ...   AGE   IP
dbserver-0   1/1     Running   ...   10s   10.8.0.112
dbserver-1   1/1     Running   ...   5s    10.8.0.113
```

Press **Ctrl**+**C** to end the `watch` command after the pods display `Running` in the `STATUS` column.

List the `mysql-svc` service endpoints, which correspond to the pods' IP addresses.

```
[student@workstation ~]$ oc get endpoints
NAME        ENDPOINTS                          AGE
mysql-svc   10.8.0.112:3306,10.8.0.113:3306   10m
```

The endpoint IPs are different on your system.

Connect to the `dbserver-1` pod by using its unique DNS name.

```
[student@workstation ~]$ oc rsh dbserver-1 \
  curl -v telnet://dbserver-1.mysql-svc:3306
* Rebuilt URL to: telnet://dbserver-1.mysql-svc:3306/
*   Trying 10.8.0.113...
* TCP_NODELAY set
* Connected to dbserver-1.mysql-svc (10.8.0.113) port 3306 (#0)
...output omitted...
```

The DNS name resolves IP that corresponds to the `dbserver-1` pod that is listed in the previous command.

7. Confirm that each instance from the `dbserver` stateful set has a persistent volume claim. Add data to each stateful set pod.

   Verify that the two persistent volume claims display `Bound` in the `STATUS` column.

```
[student@workstation ~]$ oc get pvc -l app=database
NAME                STATUS  ...  CAPACITY    ACCESS MODE  ...
data-dbserver-0     Bound   ...  1Gi         RWO          ...
data-dbserver-1     Bound   ...  1Gi         RWO          ...
```

   Verify that the `dbserver-0` pod mounts the `data-dbserver-0` PVC.

```
[student@workstation ~]$ oc get pod dbserver-0 -o json | \
  jq .spec.volumes[0].persistentVolumeClaim.claimName
"data-dbserver-0"
```

   Verify that the `dbserver-1` pod mounts the `data-dbserver-1` PVC.

```
[student@workstation ~]$ oc get pod dbserver-1 -o json | \
  jq .spec.volumes[0].persistentVolumeClaim.claimName
"data-dbserver-1"
```

   Create a table named `items` within the `sakila` database in the `dbserver-0` pod.

```
[student@workstation ~]$ oc exec -it pod/dbserver-0 -- /bin/bash -c \
  "mysql -uredhat -predhat123 sakila -e 'create table items (count INT);'"
mysql: [Warning] Using a password on the command line interface can be insecure.
```

   Create a table named `inventory` within the `sakila` database in the `dbserver-1` pod.

```
[student@workstation ~]$ oc exec -it pod/dbserver-1 -- /bin/bash -c \
  "mysql -uredhat -predhat123 sakila -e 'create table inventory (count INT);'"
mysql: [Warning] Using a password on the command line interface can be insecure.
```

8. No data replication is configured at the application level for the `dbserver` stateful set. Verify that each database pod contains unique data.

   Confirm that the `dbserver-0` pod has a database named `sakila` with one table named `items`.

```
[student@workstation ~]$ oc exec -it pod/dbserver-0 -- /bin/bash -c \
  "mysql -uredhat -predhat123 sakila -e 'show tables;'"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------------+
| Tables_in_sakila |
+------------------+
| items            |
+------------------+
```

   Confirm that the `dbserver-1` pod has a database named `sakila` with one table named `inventory`.

```
[student@workstation ~]$ oc exec -it pod/dbserver-1 -- /bin/bash -c \
  "mysql -uredhat -predhat123 sakila -e 'show tables;'"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------------+
| Tables_in_sakila |
+------------------+
| inventory        |
+------------------+
```

9. Delete a pod in the `dbserver` stateful set. A new stateful set pod is created with the same ordinal index, and the pod mounts its corresponding PVC. The data remains in the PVC and it is accessible from the pod.

   Delete the `dbserver-0` pod in the `dbserver` stateful set.

```
[student@workstation ~]$ oc delete pod dbserver-0
pod "dbserver-0" deleted
```

   Verify that a new `dbserver-0` pod is created.

```
[student@workstation ~]$ oc get pods -l app=database

NAME         READY   STATUS    RESTARTS   AGE
dbserver-0   1/1     Running   0          4s
dbserver-1   1/1     Running   0          15m
```

   Verify that the `dbserver-0` pod mounts the `data-dbserver-0` PVC.

```
[student@workstation ~]$ oc get pod dbserver-0 -o json | \
  jq .spec.volumes[0].persistentVolumeClaim.claimName
"data-dbserver-0"
```

   Verify that the new `dbserver-0` pod has the `items` table within the `sakila` database.

```
[student@workstation ~]$ oc exec -it pod/dbserver-0 -- /bin/bash -c \
  "mysql -uredhat -predhat123 sakila -e 'show tables;'"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------------+
| Tables_in_sakila |
+------------------+
| items            |
+------------------+
```

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-statefulsets
```