Guided Exercise: Kubernetes Pod and Service Networks

Deploy a database server and access it through a Kubernetes service.

Outcomes

Deploy a database server, and access it indirectly through a Kubernetes service, and also directly pod-to-pod for troubleshooting.

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. It also creates the deploy-services project and the `/home/student/DO180/labs/deploy-services/resources.txt` file. The `resources.txt` file contains commands that you can copy and paste to use in this exercise.

```
[student@workstation ~]$ lab start deploy-services
```

> **NOTE**
>
> It is safe to ignore pod security warnings for exercises in this course. OpenShift uses the Security Context Constraints controller to provide safe defaults for pod security.

Instructions

1. Log in to the OpenShift cluster as the `developer` user with `developer` as the password. Use the `deploy-services` project.

   Log in to the OpenShift cluster.

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

   Set the `deploy-services` project as the active project.

   ```
   [student@workstation ~]$ oc project deploy-services
   ...output omitted...
   ```

2. Use the `registry.ocp4.example.com:8443/rhel8/mysql-80` container image to create a MySQL deployment named `db-pod`. Add the missing environment variables for the pod to run.

   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=mypa55w0rd \
     MYSQL_DATABASE=items
   deployment.apps/db-pod updated
   ```

   Confirm that the pod is running.

   ```
   [student@workstation ~]$ oc get pods
   NAME                       READY   STATUS    RESTARTS   AGE
   db-pod-6ccc485cfc-vrc4r    1/1     Running   0          2m30s
   ```

   Your pod name might differ from the previous output.

3. Expose the `db-pod` deployment to create a ClusterIP service.

   View the deployment for the pod.

```
[student@workstation ~]$ oc get deployment
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
db-pod   1/1     1            1           3m36s
```

Expose the `db-pod` deployment to create a service.

```
[student@workstation ~]$ oc expose deployment/db-pod
service/db-pod exposed
```

4. Validate the service. Verify that the service selector matches the pod label. Then, confirm that the `db-pod` service endpoint matches the pod IP address.

   Identify the selector for the `db-pod` service. Use the `oc get service` command with the `-o wide` option.

   ```
   [student@workstation ~]$ oc get service db-pod -o wide
   NAME     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE    SELECTOR
   db-pod   ClusterIP   172.30.108.92   <none>        3306/TCP   108s   app=db-pod
   ```

   The selector shows an `app=db-pod` key:value pair.

   Capture the pod name in a variable.

   ```
   [student@workstation ~]$ PODNAME=$(oc get pods \
     -o jsonpath='{.items[0].metadata.name}')
   ```

   Query the label on the pod.

   ```
   [student@workstation ~]$ oc get pod $PODNAME --show-labels
   NAME                     READY   STATUS    RESTARTS   AGE     LABELS
   db-pod-6ccc485cfc-vrc4r  1/1     Running   0          6m50s   app=db-pod ...
   ```

   The label list includes the `app=db-pod` key-value pair, which matches the service selector.

   Retrieve the endpoints for the `db-pod` service.

   ```
   [student@workstation ~]$ oc get endpoints
   NAME     ENDPOINTS        AGE
   db-pod   10.8.0.85:3306   4m38s
   ```

   The endpoint IP is different in your output.

   Verify that the service endpoint matches the pod IP address. Use the `oc get pods` command with the `-o wide` option.

   ```
   [student@workstation ~]$ oc get pods -o wide
   NAME                      READY   STATUS    RESTARTS   AGE   IP          ...
   db-pod-6ccc485cfc-vrc4r   1/1     Running   0          54m   10.8.0.85 ...
   ```

   The service endpoint resolves to the pod's IP address.

5. Delete and re-create the `db-pod` deployment. Confirm that the `db-pod` service endpoint automatically resolves to the new pod's IP address.

   Delete the `db-pod` deployment.

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

   Verify that the service still exists without the deployment.

   ```
   [student@workstation ~]$ oc get service
   NAME     TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
   db-pod   ClusterIP   172.30.108.92  <none>        3306/TCP   9m53s
   ```

   Confirm that the service endpoints list is empty.

   ```
   [student@workstation ~]$ oc get endpoints
   NAME     ENDPOINTS   AGE
   db-pod   <none>      12m
   ```

   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=mypa55w0rd \
  MYSQL_DATABASE=items
deployment.apps/db-pod updated
```

Confirm that the new pod has the `app=db-pod` selector.

```
[student@workstation ~]$ oc get pods --selector app=db-pod -o wide
NAME                    READY   STATUS    RESTARTS   AGE   IP          ...
db-pod-6ccc485cfc-l2x   1/1     Running   0          32s   10.8.0.85 ...
```

The pod name might differ, and the pod IP address might also change.

Confirm that the endpoints include the new pod.

```
[student@workstation ~]$ oc get endpoints
NAME     ENDPOINTS        AGE
db-pod   10.8.0.85:3306   16m
```

6. Create a pod to identify the available DNS name assignments for the service.

   Create a pod named `shell` for troubleshooting. Use the `oc run` command with the `registry.ocp4.example.com:8443/openshift4/network-tools-rhel8` image.

```
[student@workstation ~]$ oc run shell -it \
  --image registry.ocp4.example.com:8443/openshift4/network-tools-rhel8
If you don't see a command prompt, try pressing enter.
bash-4.4$
```

   View the `/etc/resolv.conf` file from the `shell` pod to identify the cluster-domain name.

```
bash-4.4$ cat /etc/resolv.conf
search deploy-services.svc.cluster.local svc.cluster.local ...
nameserver 172.30.0.10
options ndots:5
```

   The container uses the `search` directive values as suffixes for DNS queries. The cluster-domain name follows `svc`.

   Test the available DNS names with the `nc` and `echo` commands.

```
bash-4.4$ nc -z db-pod.deploy-services 3306 && \
  echo "Connection success to db-pod.deploy-services:3306" || \
  echo "Connection failed"
Connection success to db-pod.deploy-services:3306
```

   Exit the interactive session.

```
bash-4.4$ exit
Session ended, resume using 'oc attach shell -c shell -i -t' command when the pod is running
```

   Delete the `shell` pod.

```
[student@workstation ~]$ oc delete pod shell
pod "shell" deleted
```

7. Test pod communications across namespaces with a new project.

   Create a second namespace with the `oc new-project` command.

```
[student@workstation ~]$ oc new-project deploy-services-2
Now using project "deploy-services-2" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

Test DNS name access from a pod in the new namespace.

```
[student@workstation ~]$ oc run shell -it --rm \
  --image registry.ocp4.example.com:8443/openshift4/network-tools-rhel8 \
  --restart Never -- nc -z db-pod.deploy-services.svc.cluster.local 3306 && \
  echo "Connection success to db-pod.deploy-services.svc.cluster.local:3306" || \
  echo "Connection failed"
pod "shell" deleted
Connection success to db-pod.deploy-services.svc.cluster.local:3306
```

Return to the `deploy-services` project.

```
[student@workstation ~]$ oc project deploy-services
Now using project "deploy-services" on server "https://api.ocp4.example.com:6443".
```

8. Use a Kubernetes job to add initialization data to the database.

   Create a job named `mysql-init` by using the `registry.ocp4.example.com:8443/redhattraining/do180-dbinit:v1` image. This image, which is based on `mysql-80`, includes a script to add initial records.

```
[student@workstation ~]$ oc create job mysql-init \
  --image registry.ocp4.example.com:8443/redhattraining/do180-dbinit:v1 \
  -- /bin/bash -c "mysql -uuser1 -pmypa55w0rd --protocol tcp \
  -h db-pod -P3306 items </tmp/db-init.sql"
job.batch/mysql-init created
```

   The `-h` option directs the command to the `db-pod` service short name. The `--` option separates `oc` arguments from the pod command. The `/tmp/db-init.sql` file, which is included in the image, contains the following queries:

```
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);
```

   Confirm the `mysql-init` job status, and wait for its completion.

```
[student@workstation ~]$ oc get job
NAME         STATUS     COMPLETIONS   DURATION   AGE
mysql-init   Complete   1/1           28s        42s
```

   Check the `mysql-init` pod status.

```
[student@workstation ~]$ oc get pods
NAME                       READY   STATUS      RESTARTS      AGE
db-pod-6ccc485cfc-2lklx    1/1     Running     0             4h24m
mysql-init-ln9cg           0/1     Completed   0             23m
```

   Delete the `mysql-init` job.

```
[student@workstation ~]$ oc delete job mysql-init
job.batch "mysql-init" deleted
```

   Verify that the `mysql-init` pod is deleted.

```
[student@workstation ~]$ oc get pods
NAME                       READY   STATUS    RESTARTS      AGE
db-pod-6ccc485cfc-2lklx    1/1     Running   0             4h2
```

9. Create a `query-db` pod to query the database service.

   Create the `query-db` pod. Use the MySQL client to query the `db-pod` service.

```
[student@workstation ~]$ oc run query-db -it --rm \
  --image registry.ocp4.example.com:8443/redhattraining/do180-dbinit:v1 \
  --restart Never \
  -- mysql -uuser1 -pmypa55w0rd --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
```

10. Use pod-to-pod communication for troubleshooting.

    Confirm the IP address of the MySQL database pod.

```
[student@workstation ~]$ oc get pods -o wide
NAME                      READY  STATUS     RESTARTS     AGE  IP          ...
db-pod-6ccc485cfc-2lklx   1/1    Running    0            4h5  10.8.0.69   ...
```

    Your pod IP address might differ.

    Capture the IP address in an environment variable.

```
[student@workstation ~]$ POD_IP=$(oc get pod -l app=db-pod \
  -o jsonpath='{.items[0].status.podIP}')
```

    Create a test pod named `shell`. Use `nc` to test the `$POD_IP` and port `3306`.

```
[student@workstation ~]$ oc run shell --env POD_IP=$POD_IP -it --rm \
  --image registry.ocp4.example.com:8443/openshift4/network-tools-rhel8 \
  --restart Never \
  -- nc -z $POD_IP 3306 && echo "Connection success to $POD_IP:3306" || \
  echo "Connection failed"
pod "shell" deleted
Connection success to 10.8.0.69:3306
```

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 deploy-services
```