Guided Exercise: Find and Inspect Container Images

Use a supported MySQL container image to run server and client pods in Kubernetes; also test two community images and compare their runtime requirements.

Outcomes

- Locate and run container images from a container registry.
- Inspect remote container images and container logs.
- Set environment variables and override entry points for a container.
- Access files and directories within a container.

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 pods-images
```

Instructions

1. Log in to the OpenShift cluster and create the `pods-images` project.

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

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

   Create the `pods-images` project.

   ```
   [student@workstation ~]$ oc new-project pods-images
   ...output omitted...
   ```

2. Authenticate to `registry.ocp4.example.com:8443`, which is the classroom container registry. This private registry hosts certain copies and tags of community images from Docker and Bitnami, as well as some supported images from Red Hat. Use `skopeo` to log in as the `developer` user, and then retrieve a list of available tags for the `registry.ocp4.example.com:8443/redhattraining/docker-nginx` container repository.

   Use the `skopeo login` command to log in as the `developer` user with the `developer` password.

   ```
   [student@workstation ~]$ skopeo login registry.ocp4.example.com:8443
   Username: developer
   Password: developer
   Login Succeeded!
   ```

   The classroom registry contains a copy and specific tags of the `docker.io/library/nginx` container repository. Use the `skopeo list-tags` command to retrieve a list of available tags for the `registry.ocp4.example.com:8443/redhattraining/docker-nginx` container repository.

   ```
   [student@workstation ~]$ skopeo list-tags \
     docker://registry.ocp4.example.com:8443/redhattraining/docker-nginx
   {
       "Repository": "registry.ocp4.example.com:8443/redhattraining/docker-nginx",
       "Tags": [
           "1.23",
           "1.23-alpine",
           "1.23-alpine-perl",
           "1.23-perl"
           "latest"
       ]
   }
   ```

3. Create a `docker-nginx` pod from the `registry.ocp4.example.com:8443/redhattraining/docker-nginx:1.23` container image. Investigate any pod failures.

   Use the `oc run` command to create the `docker-nginx` pod.

```
[student@workstation ~]$ oc run docker-nginx \
  --image registry.ocp4.example.com:8443/redhattraining/docker-nginx:1.23
pod/docker-nginx created
```

After a few moments, verify the status of the `docker-nginx` pod.

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS  RESTARTS    AGE
docker-nginx   0/1     Error   2 (23s ago) 4s
```

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS             RESTARTS      AGE
docker-nginx   0/1     CrashLoopBackOff   3 (29s ago)   38s
```

The `docker-nginx` pod failed to start.

Investigate the pod failure. Retrieve the logs of the `docker-nginx` pod to identify a possible cause of the pod failure.

```
[student@workstation ~]$ oc logs docker-nginx
...output omitted...
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/08/24 18:51:45 [warn] 1#1: the "user" directive makes sense only if the master process runs with super-user privilege
s, ignored in /etc/nginx/nginx.conf:2
nginx: [warn] the "user" directive makes sense only if the master process runs with super-user privileges, ignored in /et
c/nginx/nginx.conf:2
2025/08/24 18:51:45 [emerg] 1#1: mkdir() "/var/cache/nginx/client_temp" failed (13: Permission denied)
nginx: [emerg] mkdir() "/var/cache/nginx/client_temp" failed (13: Permission denied)
```

The pod failed to start because of permission issues for the `nginx` directories.

Create a debug pod for the `docker-nginx` pod.

```
[student@workstation ~]$ oc debug pod/docker-nginx
Starting pod/docker-nginx-debug-jf8h9 ...
Pod IP: 10.8.0.28
If you don't see a command prompt, try pressing enter.

$
```

From the debug pod, verify the permissions of the `/etc/nginx` and `/var/cache/nginx` directories.

```
$ ls -la /etc/ | grep nginx
drwxr-xr-x. 3 root root     132 Dev 14 2022 nginx
```

```
$ ls -la /var/cache | grep nginx
drwxr-xr-x. 2 root root   6 Dev 13 2022 nginx
```

Only the `root` user has permission to the `nginx` directories. The pod must therefore run as the privileged `root` user to work.

Retrieve the user ID (UID) of the `docker-nginx` user to determine whether the user is a privileged or unprivileged account. Then, exit the debug pod.

```
$ whoami
1000770000
```

```
$ exit

Removing debug pod ...
```

Your UID value might differ from the previous output.

A UID over `0` means that the container's user is a `non-root` account. Recall that OpenShift default security policies prevent regular user accounts, such as the `developer` user, from running pods and their containers as privileged accounts.

Confirm that the `docker-nginx:1.23` image requires the `root` privileged account. Use the `oc image info` command to view the configuration for the image.

```
[student@workstation ~]$ oc image info \
  registry.ocp4.example.com:8443/redhattraining/docker-nginx:1.23

Name:          registry.ocp4.example.com:8443/redhattraining/docker-nginx:1.23
Digest:        sha256:d586...82ab
Media Type:    application/vnd.docker.distribution.manifest.v2+json
Created:       2y ago
Image Size:    56.89MB in 6 layers
Layers:        31.41MB sha256:025c...84bb
               25.47MB sha256:ec0f...9437
               627B    sha256:cc9f...c9be
               958B    sha256:defc...b074
               774B    sha256:8855...3b49
               1.403kB sha256:f124...c7db
OS:            linux
Arch:          amd64
Entrypoint:    /docker-entrypoint.sh
Command:       nginx -g daemon off;
Exposes Ports: 80/tcp
Environment:   PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
               NGINX_VERSION=1.23.3
               NJS_VERSION=0.7.9
               PKG_RELEASE=1~bullseye
Labels:        maintainer=NGINX Docker Maintainers <docker-maint@nginx.com>
```

The image configuration does not define USER metadata, which confirms that the image must run as the `root` privileged user.

The `docker-nginx:1-23` container image must run as the `root` privileged user. OpenShift security policies prevent regular cluster users, such as the `developer` user, from running containers as the `root` user. Delete the `docker-nginix` pod.

```
[student@workstation ~]$ oc delete pod docker-nginx
pod "docker-nginx" deleted
```

4. Create a `bitnami-mysql` pod, which uses a copy of the Bitnami community MySQL image. The image is available in the `registry.ocp4.example.com:8443/redhattraining/bitnami-mysql` container repository.

   A copy and specific tags of the `docker.io/bitnami/mysql` container repository are hosted in the classroom registry. Use the `skopeo list-tags` command to identify available tags for the Bitnami MySQL community image in the `registry.ocp4.example.com:8443/redhattraining/bitnami-mysql` container repository.

```
[student@workstation ~]$ skopeo list-tags \
  docker://registry.ocp4.example.com:8443/redhattraining/bitnami-mysql
{
    "Repository": "registry.ocp4.example.com:8443/redhattraining/bitnami-mysql",
    "Tags": [
        "8.0.28",
        "8.0.29",
        "8.0.30",
        "8.0.31",
        "latest"
    ]
}
```

   Retrieve the configuration of the `bitnami-mysql:8.0.31` container image. Use the `oc image info` command to determine whether the image requires a privileged account by inspecting the image configuration for USER metadata.

```
[student@workstation ~]$ oc image info \
  registry.ocp4.example.com:8443/redhattraining/bitnami-mysql:8.0.31
Name:          registry.ocp4.example.com:8443/redhattraining/bitnami-mysql:8.0.31
Digest:        sha256:2a5e...7725
Media Type:    application/vnd.docker.distribution.manifest.v2+json
Created:       2y ago
Image Size:    154.8MB in 2 layers
Layers:        30.9MB  sha256:f8c1...abb5
               123.9MB sha256:31da...c470
OS:            linux
Arch:          amd64
Entrypoint:    /opt/bitnami/scripts/mysql/entrypoint.sh
Command:       /opt/bitnami/scripts/mysql/run.sh
User:          1001
Exposes Ports: 3306/tcp
...output omitted...
```

   The image defines the `1001` UID, which means that the image does not require a privileged account.

Create the `bitnami-mysql` pod with the `oc run` command. Use the `registry.ocp4.example.com:8443/redhattraining/bitnami-mysql:8.0.31` container image. Then, wait a few moments and then retrieve the pod's status with the `oc get` command.

```
[student@workstation ~]$ oc run bitnami-mysql \
  --image registry.ocp4.example.com:8443/redhattraining/bitnami-mysql:8.0.31
pod/bitnami-mysql created
```

```
[student@workstation ~]$ oc get pods
NAME            READY   STATUS              RESTARTS      AGE
bitnami-mysql   0/1     CrashLoopBackoff    2 (16s ago)   42s
```

The pod failed to start.

Examine the logs of the `bitnami-mysql` pod to determine the cause of the failure.

```
[student@workstation ~]$ oc logs bitnami-mysql
mysql 16:18:00.40
mysql 16:18:00.40 Welcome to the Bitnami mysql container
mysql 16:18:00.40 Subscribe to project updates by watching https://github.com/bitnami/containers
mysql 16:18:00.40 Submit issues and feature requests at https://github.com/bitnami/containers/issues
mysql 16:18:00.40
mysql 16:18:00.41 INFO  ==> ** Starting MySQL setup **
mysql 16:18:00.42 INFO  ==> Validating settings in MYSQL_*/MARIADB_* env vars
mysql 16:18:00.42 ERROR ==> The MYSQL_ROOT_PASSWORD environment variable is empty or not set. Set the environment variable
ALLOW_EMPTY_PASSWORD=yes to allow the container to be started with blank passwords. This is recommended only for developme
nt.
```

The `MYSQL_ROOT_PASSWORD` environment variable must be set for the pod to start.

Delete and then re-create the `bitnami-mysql` pod. Specify `redhat123` as the value for the `MYSQL_ROOT_PASSWORD` environment variable. After a few moments, verify the status of the pod.

```
[student@workstation ~]$ oc delete pod bitnami-mysql
pod "bitnami-mysql" deleted
```

```
[student@workstation ~]$ oc run bitnami-mysql \
  --image registry.ocp4.example.com:8443/redhattraining/bitnami-mysql:8.0.31 \
  --env MYSQL_ROOT_PASSWORD=redhat123
pod/bitnami-mysql created
```

```
[student@workstation ~]$ oc get pods
NAME            READY   STATUS    RESTARTS   AGE
bitnami-mysql   1/1     Running   0          20s
```

The `bitnami-mysql` pod successfully started.

Determine the UID of the container user in the `bitnami-mysql` pod. Compare this value to the UID in the container image and to the UID range of the `pods-images` project.

```
[student@workstation ~]$ oc exec -it bitnami-mysql -- /bin/bash -c "whoami && id"
1000770000
uid=1000770000(1000770000) gid=0(root) groups=0(root),1000770000
```

```
[student@workstation ~]$ oc describe project pods-images
Name:                   pods-images
...output omitted...
Annotations:    openshift.io/description=
...output omitted...
                openshift.io/sa.scc.supplemental-groups=1000770000/10000
                openshift.io/sa.scc.uid-range=1000770000/10000
...output omitted...
```

Your values for the UID of the container and the UID range of the project might differ from the previous output.

The container user UID is the same as the specified UID range in the namespace. Notice that the container user UID does not match the `1001` UID of the container image. For a container to use the specified UID of a container image, the pod must be created with a privileged OpenShift user account, such as the `admin` user.

5. The private classroom registry hosts a copy of a supported MySQL image from Red Hat. Retrieve the list of available tags for the `registry.ocp4.example.com:8443/rhel9/mysql-80` container repository. Compare the `rhel9/mysql-80` container image release version that is associated with each tag.

Use the `skopeo list-tags` command to list the available tags for the `rhel9/mysql-80` container image.

```
[student@workstation ~]$ skopeo list-tags \
  docker://registry.ocp4.example.com:8443/rhel9/mysql-80
{
    "Repository": "registry.ocp4.example.com:8443/rhel9/mysql-80",
    "Tags": [
        "1-224",
        "1-224-source",
        "1-228",
        "1-228-source",
        "1-237",
              "latest",
        "1"
    ]
}
```

Several tags are available:

- The `latest` and `1` tags are floating tags, which are aliases to other tags, such as the `1-237` tag.
- The `1-228` and `1-224` tags are fixed tags, which point to a build of a container.
- The `1-228-source` and `1-224-source` tags are source containers, which provide the necessary sources and license terms to rebuild and distribute the images.

Use the `skopeo inspect` command to compare the `rhel9/mysql-80` container image release versions and SHA IDs that are associated with the identified tags.

> **NOTE**
>
> To improve readability, the instructions truncate the SHA-256 strings.
>
> On your system, the commands return the full SHA-256 strings.

```
[student@workstation ~]$ skopeo inspect \
  docker://registry.ocp4.example.com:8443/rhel9/mysql-80:latest
...output omitted...
    "Name": "registry.ocp4.example.com:8443/rhel9/mysql-80",
    "Digest": "sha256:d282...f38f",
...output omitted...
    "Labels":
...output omitted...
        "name": "rhel9/mysql-80",
        "release": "237",
...output omitted...
```

You can also format the output of the `skopeo inspect` command with a Go template. Append the template objects with `\n` to add new lines between the results.

```
[student@workstation ~]$ skopeo inspect --format \
  "Name: {{.Name}}\n Digest: {{.Digest}}\n Release: {{.Labels.release}}" \
  docker://registry.ocp4.example.com:8443/rhel9/mysql-80:latest
Name: registry.ocp4.example.com:8443/rhel9/mysql-80
 Digest: sha256:d282...f38f
 Release: 237
```

```
[student@workstation ~]$ skopeo inspect --format \
  "Name: {{.Name}}\n Digest: {{.Digest}}\n Release: {{.Labels.release}}" \
  docker://registry.ocp4.example.com:8443/rhel9/mysql-80:1
Name: registry.ocp4.example.com:8443/rhel9/mysql-80
 Digest: sha256:d282...f38f
 Release: 237
```

```
[student@workstation ~]$ skopeo inspect --format \
  "Name: {{.Name}}\n Digest: {{.Digest}}\n Release: {{.Labels.release}}" \
  docker://registry.ocp4.example.com:8443/rhel9/mysql-80:1-237
Name: registry.ocp4.example.com:8443/rhel9/mysql-80
 Digest: sha256:d282...f38f
 Release: 237
```

The `latest`, `1`, and `1-237` tags resolve to the same release versions and SHA IDs. The `latest` and `1` tags are floating tags for the `1-237` fixed tag.

6. The classroom registry hosts a copy and certain tags of the `registry.redhat.io/rhel9/mysql-80` container repository. Use the `oc run` command to create a `rhel9-mysql` pod from the `registry.ocp4.example.com:8443/rhel9/mysql-80:1-237` container image. Verify the status of the pod and then inspect the container logs for any errors.

   Create a `rhel9-mysql` pod with the `registry.ocp4.example.com:8443/rhel9/mysql-80:1-237` container image.

   ```
   [student@workstation ~]$ oc run rhel9-mysql \
     --image registry.ocp4.example.com:8443/rhel9/mysql-80:1-237
   pod/rhel9-mysql created
   ```

   After a few moments, retrieve the pod's status with the `oc get` command.

   ```
   [student@workstation ~]$ oc get pods
   NAME          READY   STATUS             RESTARTS      AGE
   bitnami-mysql 1/1     Running            0             12m
   rhel9-mysql   0/1     CrashLoopBackoff   1 (2s ago)    16s
   ```

   The pod failed to start.

   Retrieve the logs for the `rhel9-mysql` pod to determine why the pod failed.

   ```
   [student@workstation ~]$ oc logs rhel9-mysql
   => sourcing 20-validate-variables.sh ...
   You must either specify the following environment variables:
     MYSQL_USER (regex: '^[a-zA-Z0-9_]+$')
     MYSQL_PASSWORD (regex: '^[a-zA-Z0-9_~!@#$%^&*()-=<>,.?;:|]+$')
     MYSQL_DATABASE (regex: '^[a-zA-Z0-9_]+$')
   Or the following environment variable:
     MYSQL_ROOT_PASSWORD (regex: '^[a-zA-Z0-9_~!@\#$%^&*()-=<>,.?;:|]+$')
   Or both.
   Optional Settings:
     MYSQL_LOWER_CASE_TABLE_NAMES (default: 0)
   ...output omitted...
   ```

   The pod failed because the required environment variables were not set for the container.

7. Delete the `rhel9-mysql` pod. Create another `rhel9-mysql` pod and specify the necessary environment variables. Retrieve the status of the pod and inspect the container logs to confirm that the new pod is working.

   Delete the `rhel9-mysql` pod with the `oc delete` command. Wait for the pod to delete before continuing to the next step.

   ```
   [student@workstation ~]$ oc delete pod rhel9-mysql
   pod "rhel9-mysql" deleted
   ```

   Create another `rhel9-mysql` pod from the `registry.ocp4.example.com:8443/rhel9/mysql-80:1-237` container image. Use the `oc run` command with the `--env` option to specify the following environment variables and their values:

Variable	Value
`MYSQL_USER`	`redhat`
`MYSQL_PASSWORD`	`redhat123`
`MYSQL_DATABASE`	`worldx`

   ```
   [student@workstation ~]$ oc run rhel9-mysql \
     --image registry.ocp4.example.com:8443/rhel9/mysql-80:1-237 \
     --env MYSQL_USER=redhat \
     --env MYSQL_PASSWORD=redhat123 \
     --env MYSQL_DATABASE=worldx
   pod/rhel9-mysql created
   ```

   After a few moments, retrieve the status of the `rhel9-mysql` pod with the `oc get` command. View the container logs to confirm that the database on the `rhel9-mysql` pod is ready to accept connections.

   ```
   [student@workstation ~]$ oc get pods
   NAME          READY   STATUS    RESTARTS   AGE
   bitnami-mysql 1/1     Running   0          10m
   rhel9-mysql   1/1     Running   0          20s
   ```

```
[student@workstation ~]$ oc logs rhel9-mysql
...output omitted...
2025-08-26T20:14:14.333599Z 0 [System] [MY-013576] [Server] X Plugin ready for connections. Bind-address: '::' port: 3306
0, socket: /var/lib/mysql/mysqlx.sock
2025-08-26T20:14:14.333641Z 0 [System] [MY-013576] [Server] /usr/libexec/mysqld: ready for connections. Version: '8.0.30'
socket: '/var/lib/mysql/mysql.sock'  port: 3306  Source distribution.
```

The `rhel9-mysql` pod is ready to accept connections.

8. Determine the location of the MySQL database files for the `rhel9-mysql` pod. Confirm that the directory contains the `worldx` database.

   Use the `oc image` command to inspect the `rhel9/mysql-80:1-237` image in the `registry.ocp4.example.com:8443` classroom registry.

```
[student@workstation ~]$ oc image info \
  registry.ocp4.example.com:8443/rhel9/mysql-80:1-237
Name:          registry.ocp4.example.com:8443/rhel9/mysql-80:1-237
...output omitted...
Command:       run-mysqld
Working Dir:   /opt/app-root/src
User:          27
Exposes Ports: 3306/tcp
Environment:   container=oci
               STI_SCRIPTS_URL=image:///usr/libexec/s2i
               STI_SCRIPTS_PATH=/usr/libexec/s2i
               APP_ROOT=/opt/app-root
               PATH=/opt/app-root/src/bin:/opt/app-root/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
               PLATFORM=el9
               MYSQL_VERSION=8.0
               APP_DATA=/opt/app-root/src
               HOME=/var/lib/mysql
```

   The container manifest sets the `HOME` environment variable for the container user to the `/var/lib/mysql` directory.

   Use the `oc exec` command to list the contents of the `/var/lib/mysql` directory.

```
[student@workstation ~]$ oc exec -it rhel9-mysql -- ls -la /var/lib/mysql
total 12
drwxrwxr-x. 1 mysql root          102 Aug 26 16:28 .
drwxr-xr-x. 1 root  root           19 Jan 17  2023 ..
drwxrwxr-x. 1 mysql root         4096 Aug 26 20:54 data
srwxrwxrwx. 1 mysql 1000770000      0 Aug 26 16:28 mysql.sock
-rw-------. 1 mysql 1000770000      2 Aug 26 16:28 mysql.sock.lock
srwxrwxrwx. 1 mysql 1000770000      0 Aug 26 16:28 mysqlx.sock
-rw-------. 1 mysql 1000770000      2 Aug 26 16:28 mysqlx.sock.lock
```

   A `data` directory exists in the `/var/lib/mysql` directory.

   Use the `oc exec` command again to list the contents of the `/var/lib/mysql/data` directory.

```
[student@workstation ~]$ oc exec -it rhel9-mysql \
  -- ls -la /var/lib/mysql/data | grep worldx
drwxr-x---. 2 1000770000 root     6 Aug  26 16:28  worldx
```

   The `/var/lib/mysql/data` directory contains the `worldx` database with the `worldx` directory.

9. Determine the IP address of the `rhel9-mysql` pod. Next, create another MySQL pod, named `mysqlclient`, to access the `rhel9-mysql` pod. Confirm that the `mysqlclient` pod can view the available databases on the `rhel9-mysql` pod with the `mysqlshow` command.

   Identify the IP address of the `rhel9-mysql` pod.

```
[student@workstation ~]$ MYSQL_IP=$(oc get pods rhel9-mysql -o=jsonpath='{.status.podIP}') && echo $MYSQL_IP
"10.8.0.69"
```

   Note the IP address stored in the `MYSQL_IP` variable. Your IP address might differ from the previous output.

   Use the `oc run` command to create a pod named `mysqlclient` that uses the `registry.ocp4.example.com:8443/rhel9/mysql-80:1-237` container image. Set the value of the `MYSQL_ROOT_PASSWORD` environment variable to `redhat123`, and then confirm that the pod is running.

```
[student@workstation ~]$ oc run mysqlclient \
  --image registry.ocp4.example.com:8443/rhel9/mysql-80:1-237 \
  --env MYSQL_ROOT_PASSWORD=redhat123
pod/mysqlclient created
```

```
[student@workstation ~]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
bitnami-mysql 1/1     Running   0          15m
mysqlclient   1/1     Running   0          19s
rhel9-mysql   1/1     Running   0          8m
```

Use the `oc exec` command with the `-it` options to execute the `mysqlshow` command on the `mysqlclient` pod. Connect as the `redhat` user and specify the host as the IP address of the `rhel9-mysql` pod. When prompted, enter `redhat123` for the password, although no characters are shown.

```
[student@workstation ~]$ oc exec -it mysqlclient \
  -- mysqlshow -u redhat -p -h $MYSQL_IP
Enter password: redhat123
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| performance_schema |
| worldx             |
+--------------------+
```

The `worldx` database on the `rhel9-mysql` pod is accessible to the `mysql-client` pod.

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 pods-images
```