Containers: Hands-On Examples

By Kelvin Lai

This lab serves to solidify the understanding of container architecture principles covered in the course.

Document Convention:

Ellipses (...)	Due to its excessive length, some of the output produced on the command lines have been abbreviated.
Red Arrow (⬅)	Pay special attention to these lines. The trainer will explain them in detail.
INPUT	The text that is **Bold**, *Italic*, and in **Dark Green** should be completed using your specific information.

Prerequisite:

- Internet connection
- The lab uses the quay.io registry. You may use your RH ID to login to quay.io or use any container registry server account that you have access to.

Table of Contents:

Exercise 1: Registry, Image and Container

1. Check local storage for images.

```
[user@host ~]$ podman images
REPOSITORY  TAG       IMAGE ID      CREATED       SIZE

[user@host ~]$
```

2. Verify repositories configured for podman usage.

```
[user@host ~]$ podman info
  …
  slirp4netns:  ⬅
       executable: /usr/bin/slirp4netns
       package: slirp4netns-1.1.12-4.el9.x86_64
  …
Registries: ⬅
  search:
  - registry.fedoraproject.org
  - registry.access.redhat.com
  - registry.centos.org
  - quay.io
  - docker.io
store:
  configFile: /home/user/.config/containers/storage.conf
  …
  graphRoot: /home/user/.local/share/containers/storage ⬅
  …
  runRoot: /run/user/1000/containers ⬅
  volumePath: /home/user/.local/share/containers/storage/volumes
…

[user@host ~]$
```

3. Search for RHEL8 Universal Base Image.

```
[user@host ~]$ podman search ubi8
NAME                                        DESCRIPTION
registry.access.redhat.com/ubi8/go-toolset  Platform for building and running Go
1.11.5 based applications
…
registry.access.redhat.com/ubi8/ubi         Provides the latest release of the Red
Hat Universal Base Image 8
registry.access.redhat.com/ubi8             The Universal Base Image is designed
and engineered to be the base layer for all of your containerized applications, middleware
and utilities. This base image is freely redistributable, but Red Hat only supports Red
Hat technologies through subscriptions for Red Hat products. This image is maintained by
Red Hat and updated regularly.
…
quay.io/app-sre/ubi8-ubi-minimal            Mirror                                of
https://access.redhat.com/containers/#/registry.access.redhat.com/ubi8/ubi-minimal
…
docker.io/redhat/ubi8                       Red Hat Universal Base Image 8
docker.io/redhat/ubi8-minimal               Red Hat Universal Base Image 8 Minimal
docker.io/redhat/ubi8-init                  Red Hat Universal Base Image 8 Init
docker.io/redhat/ubi8-micro                 Red Hat Universal Base Image 8 Micro
…
[user@host ~]$
```

4. Download and verify the image. The download size may differ from the storage size because images are stored in a compressed format in the repository.

```
[user@host ~]$ podman pull registry.access.redhat.com/ubi8/ubi
Trying to pull registry.access.redhat.com/ubi8/ubi:latest... ⬅
Getting image source signatures
Checking if image destination supports signatures
Copying blob 57755749ebfe done
Copying config 0dc8d21c3c done
Writing manifest to image destination
Storing signatures
0dc8d21c3cb04ae0ab3b7abfd738ce966d2da69755cbeb7d97335cb224bccfe9

[user@host ~]$ podman images
REPOSITORY                          TAG           IMAGE ID      CREATED       SIZE
registry.access.redhat.com/ubi8/ubi  latest       0dc8d21c3cb0  4 days ago  215 MB

[user@host ~]$
```

5. Examine the image to identify the vendor and determine the number of layers it contains.

```
[user@host ~]$ podman inspect ubi
CONTAINER ID  IMAGE          COMMAND       CREATED       STATUS        PORTS
NAMES
```

NOTE: Since we've already downloaded the image into our local store, we could refer to it by **ubi:latest**, or just **ubi**. The 'latest' tag is optional, as it's the default if no tag is specified.

6. Verify that there are no containers running.

```
[user@host ~]$ podman ps
CONTAINER ID  IMAGE          COMMAND       CREATED       STATUS        PORTS
NAMES

[user@host ~]$
```

7. Run a command inside a container.

```
[user@host ~]$ podman run registry.access.redhat.com/ubi8/ubi:latest whoami
root ⬅

[user@host ~]$ whoami
user

[user@host ~]$
```

NOTE: We could have just used `podman run ubi whoami` here.

8. List running containers; the newly created container won't be visible as it has stopped due to the 'whoami' process exiting.

```
[user@host ~]$ podman ps
CONTAINER ID  IMAGE          COMMAND       CREATED       STATUS        PORTS
NAMES

[user@host ~]$
```

9. To list all containers, whether running or stopped, use the **-a** option.

```
[user@host ~]$ podman ps -a
CONTAINER ID  IMAGE                                         COMMAND       CREATED
STATUS                     PORTS         NAMES
bd24aa126548  registry.access.redhat.com/ubi8/ubi:latest  whoami        6    seconds    ago
Exited (0) 7 seconds ago            determined_austin    ⬅

[user@host ~]$
```

NOTE: When a container is created without a name, Podman generates a unique name by combining two dictionary words with an underscore. To reference the container, you can use the full name, such as `determined_austin` (autocompletable with TAB), or use parts of the unique container ID, like `bd`, especially when no other containers share similar IDs.

10. We no longer need this container. Please delete it, and you can use TAB completion for the name.

```
[user@host ~]$ podman rm determined_austin
bd24aa126548ba791e314ef521bbaf716521dc1dc830ba337110e38da9688692

[user@host ~]$ podman ps -a
CONTAINER ID  IMAGE          COMMAND       CREATED       STATUS        PORTS
NAMES

[user@host ~]$
```

NOTE: If you want Podman to automatically remove a container after its process has ended, you can utilize the `--rm` option. For instance, you can achieve this by running a command like: **`podman run --rm ubi date`**

Exercise 2: Creating an Image from a Container

Objectives:

- Deploying a web server in a container.
- Use *port mapping* to test the web server.
- Create an *image* from the container.

1. Create a container named “website”, map the localhost port 12345 to the container's port 8080 for later testing. Take note that the command prompt changes to display we are now inside the container.

```
[user@host ~]$ podman run --name website -p 12345:8080 -it ubi /bin/bash

[root@309a27fd3f8a /]#    ⬅
```

2. Install the Apache web server, configure it to listen to port 8080, and create a basic welcome page.

```
[root@309a27fd3f8a /]# yum install -y httpd
Updating Subscription Management repositories.
…

Complete!
[root@309a27fd3f8a /]# yum clean all
Updating Subscription Management repositories.
Unable to read consumer identity
Subscription Manager is operating in container mode.

This system is not registered with an entitlement server. You can use subscription-manager
to register.

25 files removed

[root@309a27fd3f8a /]# sed -i 's/Listen 80/Listen 8080/' /etc/httpd/conf/httpd.conf

[root@309a27fd3f8a /]# echo Hello World > /var/www/html/index.html
```

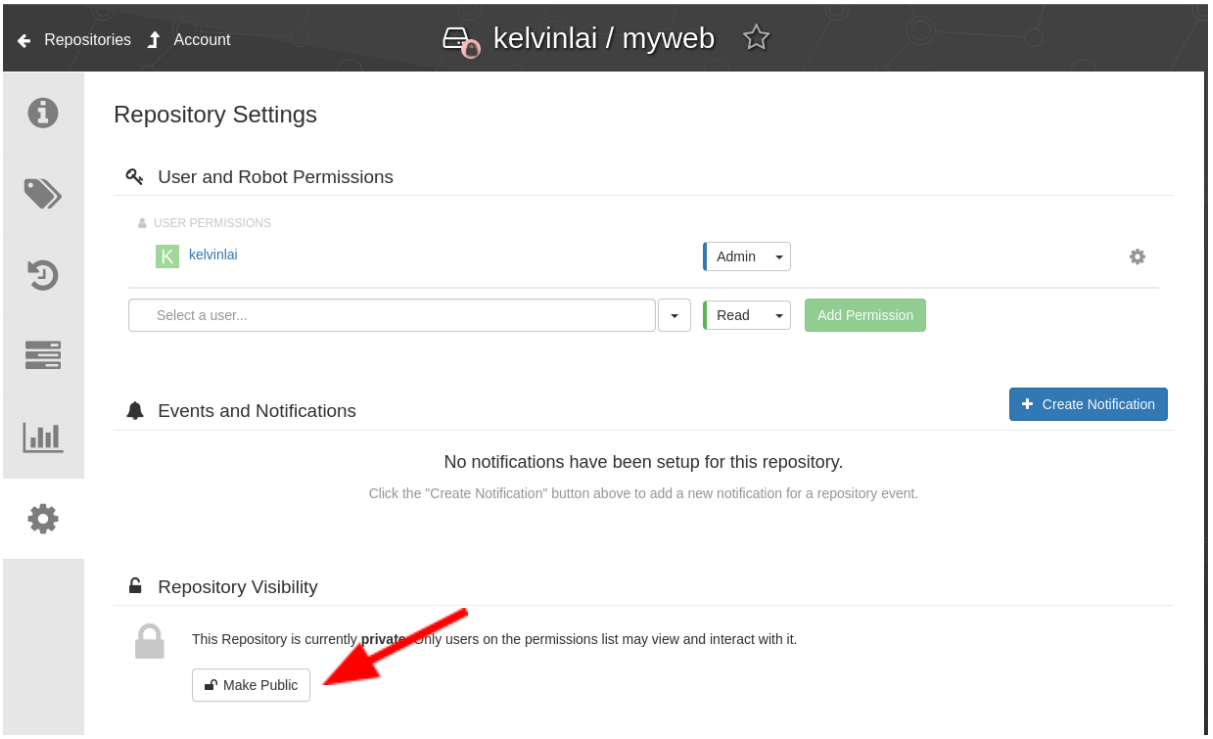
3. ***Extra steps*** needed for the container to be used in **OpenShift**.

```
[root@309a27fd3f8a /]# chgrp -R 0 /var/log/httpd /var/run/httpd

[root@309a27fd3f8a /]# chmod -R g=u /var/log/httpd /var/run/httpd
```

4. Everything is complete now. Let us save the current container image for future usage in OpenShift. Pay special attention to the naming convention used for the image, as it will be pushed to your quay.io repository later.

```
[root@309a27fd3f8a /]# exit
exit

[user@host ~]$ podman ps
CONTAINER ID  IMAGE          COMMAND       CREATED       STATUS        PORTS
NAMES

[user@host ~]$ podman ps -a
CONTAINER ID  IMAGE                                       COMMAND       CREATED
STATUS                      PORTS                 NAMES
309a27fd3f8a   registry.access.redhat.com/ubi8/ubi:latest   /bin/bash     30 minutes ago
Exited (1) 9 seconds ago  0.0.0.0:12345->8080/tcp  website   ⬅

[user@host ~]$ podman commit -a 'YOUR NAME' -c 'EXPOSE 8080' -c 'ENTRYPOINT ["httpd"]' -c
'CMD ["-D","FOREGROUND"]' website quay.io/YOUR RH ID/myweb:1.0
Getting image source signatures
Copying blob e36062b9b3d0 skipped: already exists
Copying blob 187b698fcfc8 done
Copying config b66ce0256e done
Writing manifest to image destination
Storing signatures
b66ce0256e61cc5873ffc6645c7dad2e98848661ea7898ec3a4f27c12c55df83

[user@host ~]$ podman images
REPOSITORY                          TAG           IMAGE ID      CREATED        SIZE
quay.io/kelvinlai/myweb             1.0           b66ce0256e61  5 seconds ago  237 MB
registry.access.redhat.com/ubi8/ubi  latest       0dc8d21c3cb0  5 days ago     215 MB

[user@host ~]$ podman login quay.io
Username: YOUR USER ID
Password: YOUR PASSWORD
Login Succeeded!

[user@host ~]$ podman push quay.io/YOUR RH ID/myweb:1.0
Getting image source signatures
Copying blob b77db3cfba63 done
Copying blob e36062b9b3d0 done
Copying config f26dd4ba74 done
Writing manifest to image destination
Storing signatures

[user@host ~]$
```

5. Login to quay.io and make the image public for future usage in OpenShift.



6. Test the container.
    a. Test localhost port 12345. This will fail because the container has already stopped.

```
[user@host dummy]$ curl localhost:12345
curl: (7) Failed to connect to localhost port 12345 after 0 ms: Couldn't connect to server

[user@host dummy]$
```

    b. Start the container that has exited and start the apache web server.

```
[root@host ~]# podman start -a website
[root@309a27fd3f8a /]# apachectl -k start
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
using 10.0.2.100. Set the 'ServerName' directive globally to suppress this message
```

    c. Open another terminal and test the localhost port 12345.

```
[user@host dummy]$ curl localhost:12345
Hello World

[user@host dummy]$
```

d. Switch back to the first terminal and stop the apache web server.

```
[root@309a27fd3f8a /]# apachectl -k stop
Passing arguments to httpd using apachectl is no longer supported.
You can only start/stop/restart httpd using this script.
If you want to pass extra arguments to httpd, edit the
/etc/sysconfig/httpd config file.
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
using 10.0.2.100. Set the 'ServerName' directive globally to suppress this message

[root@309a27fd3f8a /]#
```

e. Use the second terminal to verify the service is down.

```
[user@host ~]$ curl localhost:12345
curl: (56) Recv failure: Connection reset by peer

[user@host ~]$
```

f. Exit

```
[root@309a27fd3f8a /]# exit
exit
[user@host ~]$ podman ps -a
CONTAINER ID  IMAGE                                         COMMAND       CREATED
STATUS                     PORTS                 NAMES
309a27fd3f8a  registry.access.redhat.com/ubi8/ubi:latest  /bin/bash     30 minutes ago
Exited (1) 9 seconds ago  0.0.0.0:12345->8080/tcp  website
```

NOTE: In practice, it's not common to save an image before testing. This exercise is designed this way to ensure the image's usability in later OpenShift lessons. In real-world scenarios, we often use a *Containerfile* to create an image.

Exercise 3: Persistent Storage

Objectives:

- Use the newly created image to deploy a container.
- Attach local storage to a container generated from the new image.

In this exercise, we will utilize a bind mount. It's important to note that volume and tmpfs mounting is not showcased in this particular exercise.

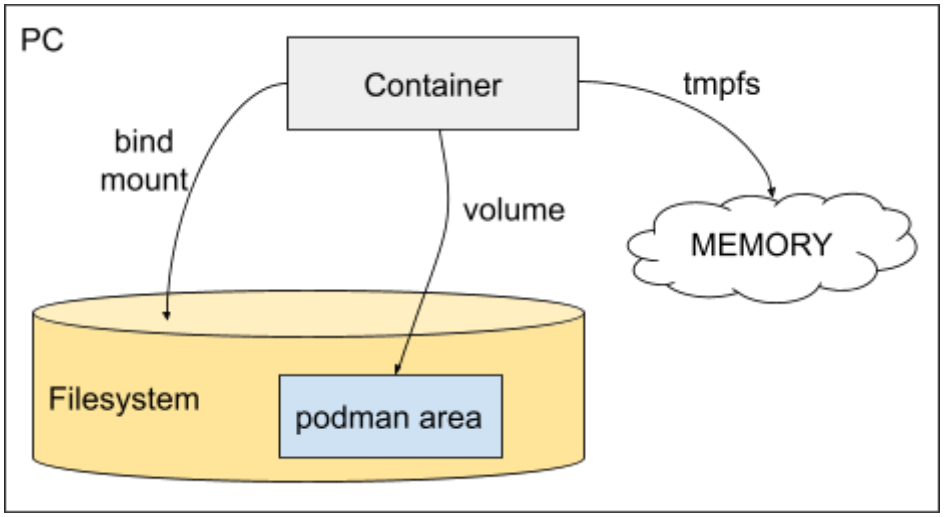


Diagram 1: bind mount vs volume vs tmpfs mount

1. Create and populate directory to be used for persistent data

```
[user@host ~]$ cd

[user@host ~]$ mkdir web_data

[user@host ~]$ echo Welcome to my website. > website_data/index.html
```

2. Do the following:
   a. Start a new container named new_website
   b. Bind mount /home/user/website to web server's DocumentRoot at /var/www/html
   c. Listen to localhost port 12346
   d. Run the container in the background
   e. When the process terminates, auto delete the container.

```
[user@host ~]$ podman run -d --rm -p 12346:8080 -v /home/user/web_data:/var/www/html:z
--name new_website myweb:1.0
9de3878378cbad067d1380269149f36314370af3f450c9a9ea82a0c1495b0443

[user@host ~]$ podman ps
CONTAINER ID  IMAGE                          COMMAND        CREATED        STATUS
PORTS                 NAMES
9de3878378cb  quay.io/kelvinlai/myweb:1.0   -D FOREGROUND  2 seconds ago  Up 2 seconds ago
0.0.0.0:12346->8080/tcp  new_website
```

3. Test the container

```
[user@host ~]$ curl localhost:12346
Welcome to my website.

[user@host ~]$ echo Success >> web_data/index.html

[user@host ~]$ curl localhost:12346
Welcome to my website.
Success
```

4. Getting into an existing container for troubleshooting or checking.

```
[user@host ~]$ podman exec -it new_website /bin/bash
new_website

[root@309a27fd3f8a /]# ls
index.html

[root@309a27fd3f8a /]# exit

[user@host ~]$
```

5. Cleanup

```
[user@host ~]$ podman stop new_website
new_website

[user@host ~]$ podman ps -a
CONTAINER ID  IMAGE                                         COMMAND       CREATED
       STATUS                     PORTS                 NAMES
309a27fd3f8a  registry.access.redhat.com/ubi8/ubi:latest  /bin/bash     About an hour ago
Exited (1) 35 minutes ago  0.0.0.0:12345->8080/tcp  website

[user@host ~]$ podman rm website
309a27fd3f8aaba232887af2d6c8c74343d058a8d3a64b281fb93130ba9627a0
```