Lab 2

Chapter 2

Outline

- After completing this section, students should be able to:
    - Search for and fetch container images with Podman.
    - Run and configure containers locally.
    - Use the Red Hat Container Catalog.

Definitions

An image is a file-system bundle that contains all dependencies required to execute a process: files in the file system, installed packages, available resources, running processes, and kernel modules.

Docker images are also immutable. While they can't be changed, they can be duplicated, shared or deleted

A container is an instance of an image, with known lifecycle start,run, pause, stop,delete

Applications can **run inside containers**.

Running a containerized application **requires a container image**.

An image provides **all** application **files**, **libraries**, and **dependencies** the application **needs** to run.

Where can we find images?

Images can be found in image registries.

The **registry** acts as a **place to store container images** and share them out via a process of **uploading** to (**pushing**) and **downloading** from (**pulling**).

Podman

- Podman is a tool for creating, deploying, and running applications easily.
- **It allows us to package our applications with all the dependencies, and distribute them as individual bundles**.
- Podman guarantees that our application will run in the same way on every podman instance.
- Podman follows the same command patterns as the Docker CLI, so there is no need to learn a new toolset.
- Podman is compatible with Kubernetes. Kubernetes can use Podman to manage its containers.

Podman command to search for image:

```
sudo podman search {image name}
```

Ex: **sudo podman search mysql**

After you have found an image, you can use Podman to **download** it. When using the **pull** command,

Podman fetches the image and saves it **locally** for future use

```
sudo podman pull {image-name}

sudo podman pull quay.io/bitnami/mongodb
```

Images naming

Container images are named based on the following syntax:

registry_name/user_name/image_name:tag

- First **registry_name**, **the name of the registry storing the image**(docker.io, quay.io, ..)
- **user_name** stands for t**he user or organization the image belongs to**.
- The **image_name** should be **unique** in user namespace.
- The tag identifies the image version. If the image name includes no image tag, **latest is assumed.**

Run an image

The podman run command **runs a container locally based on an image**. At a minimum, the command requires t**he name of the image to execute in the container.**

Inspect a container

Show All info about the latest running container

```
sudo podman inspect -l
```

Or inspect specific container

```
sudo podman inspect my-httpd-container
```

To fetch certain info use -f

```
sudo podman inspect -l -f "{{.NetworkSettings.IPAddress}}"
```

Podman run command may contain more options

1- use **-d** for detach mode to run a container as a background service

Podman returns the container ID on the screen, allowing you to continue to run commands in the same terminal while the container runs in the background:

```
sudo podman run -d rhscl/httpd-24-rhel7:2.4-36.8
```

2- use -it to starts a Bash terminal inside the container(view file system)

```
sudo podman run -it ubi7/ubi:7.7 /bin/bash
```

3- we can pass environment variable using -e option. Check the image page on the repo to know its env ex: https://hub.docker.com/_/mongo

```
sudo podman run --name mysql-custom \
> -e MYSQL_USER=redhat -e MYSQL_PASSWORD=r3dh4t \
> -d rhmap47/mysql:5.5
```

The podman ps command displays the container ID and names for **all actively running containers:**

sudo podman ps

```
CONTAINER ID    IMAGE                    COMMAND                   ... NAMES
 47c9aad6049   rhscl/httpd-24-rhel7  "httpd -D FOREGROUND"  ... focused_fermat
```

Container ID and name are generated automatically

You can specify a custom name using

sudo podman run --name my-httpd-container rhscl/httpd-24-rhel7

```
sudo podman ps -a
```

List all local containers including the stopped ones

Stop a container **: sudo podman stop my-httpd-container**

Restart a container**: sudo podman restart my-httpd-container**

Remove one container**: sudo podman rm my-httpd-container**

Remove all containers**: sudo podman rm -a**

Stop all container**: sudo podman stop -a**

Container catalogs

https://catalog.redhat.com/software/containers/search

https://hub.docker.com/

Guided Exercise: Creating a MySQL Database Instance

- https://rha.ole.redhat.com/rha/app/courses/do180-4.10/52f11f0e-a277-4441-9d11-e3d56d7defca/pages/ch02s02

- [student@workstation ~]$ lab container-create start

Root and Rootless containers

- Historically, tools for container creation required that runtime engines run as root → **Root containers**
- A better practice is to containerize applications in such a way that they do not require a privileged user to run. These applications should use a well-known user instead → **Rootless containers**

Rootless Containers

- Advantages and Disadvantages →Brainstorming
- Rootless containers use the User namespace to make application code appear to be running as root. However, from the host's perspective, permissions are limited to those of a regular user.

Guided Exercise: Exploring Root and Rootless Containers

- https://rha.ole.redhat.com/rha/app/courses/do180-4.10/52f11f0e-a277-4441-9d11-e3d56d7defca/pages/ch02s04
- [student@workstation ~]$ lab container-rootless start

Lab Task: Creating Containerized Services

https://rha.ole.redhat.com/rha/app/courses/do180-4.10/52f11f0e-a277-4441-9d11-e3d56d7defca/pages/ch02s05

```
[student@workstation ~]$ lab container-review start
```