Podman (Docker), and Singularity (Apptainer) HPC Guide

What is a Container?

Containers are portable, self-contained units that package software and its dependencies, facilitating consistent deployment across diverse computing environments. They encapsulate applications and necessary components, ensuring reliable execution in various settings.

For example, consider a web application that requires specific versions of Python and a database. Instead of worrying about system compatibility issues, you can create a container for the web application with its dependencies. This container can then be run on any system that supports containerization, ensuring a consistent and reliable environment for the application to execute.

How to Implement Containers in HPC?

Podman (Docker), and Singularity (Apptainer)

Docker, Podman, and Singularity are containerization tools that simplify software deployment. Docker is a widely-used platform for creating and managing containers. Podman is a secure alternative to Docker, offering compatibility with the Docker CLI (Command Line Interface). Singularity is designed for high-performance computing environments, emphasizing reproducibility and compatibility with existing workflows. Each tool streamlines the process of packaging and running applications in isolated environments for portability and consistency.

On cluster RED (HPC), use Podman (Docker) to build image and Apptainer (Singularity) for running containers. Podman is a secure Docker alternative, interchangeable with the standard Docker CLI on the RED cluster.

Docker

Dockerfile --

Docker provides a straightforward "Dockerfile" to define the image's configuration. The Docker CLI then executes commands to build the image using this file.

Download and install Docker Desktop for your operating system from the official Docker website (https://www.docker.com/products/docker-desktop)
or use it in HPC, following the demonstration in HPC.

Example:

1. Create a Dockerfile and an `app/input` folder containing the hello.txt “hello, running Singularity with the file provided” in the current same directory. Dockerfile -- use text editor to create it but don’t save it as .txt; save without file extension.

Dockerfile:

```
# Use the official Ubuntu image as the base
FROM ubuntu:latest

# Set the working directory
WORKDIR /app

# Copy files (including hello.txt) into the container
COPY app/input /app/input
```

```
# Make sure the file is readable by all users
RUN chmod a+r /app/input/hello.txt

# Set the default command to run when the container starts
CMD cat /app/input/hello.txt
```

---

app/input folder:

It contains `hello.txt` file for running with the command “cat /app/input/hello.txt” as described in the Dockerfile.

2. Build image:

Command –

```
docker build -t sample_app .
```

```
80027256@red2 podman_example]$ docker build -t sample_app .
Successfully tagged localhost/sample_app:latest
```

The Image “sample_app” is built.

3. Run the container:

Command –

```
docker run -it sample_app:latest
```

```
80027256@red2 podman_example]$ docker run -it sample_app:latest
hello, running singualrity with file provided
```

The terminal displays “hello, running Singularity with the file provided” as same in `hello.txt`.

Podman

Podmanfile --

Podman: Podman offers a similar approach to Docker, utilizing a Docker-compatible CLI for building images. Users can specify a container configuration using a Podmanfile or Dockerfile. Podman is used in HPC instead of Docker in the OS.

Note: Running podman needs to be on HPC root, not on a compute node.

Example:

1. Create a podmanfile and an `app/input` folder containing the hello.txt "hello, running Singularity with the file provided" in the current same directory. Podmanfile -- use text editor to create it but don't save it as .txt; save without file extension.

Podmanfile:

(Same content as Dockerfile in the Docker example)

```
# Use the official Ubuntu image as the base
FROM ubuntu:latest

# Set the working directory
WORKDIR /app

# Copy files (including hello.txt) into the container
COPY app/input /app/input

# Make sure the file is readable by all users
```

```
RUN chmod a+r /app/input/hello.txt

# Set the default command to run when the container starts
CMD cat /app/input/hello.txt
```

---

app/input folder:

(Same content as in the Docker example)
It contains `hello.txt` file for running with the command “cat /app/input/hello.txt” as described in the Dockerfile.

2. Build image:
Command –

```
podman build -t sample_app .
```

```
80027256@red2 podman_example]$ podman build -t sample_app .
Successfully tagged localhost/sample_app:latest
```

The Image “sample_app” is built.

Check image list:

Command –

```
podman images
```

It shows the image list.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
localhost/sample_app	latest	0c1e914e51db	8 seconds ago	80.4 MB

---

3. Run the container:

Command –

```
podman run -it localhost/sample_app
```

80027256@red2 podman_example]$ podman run -it sample_app:latest
hello, running singualrity with file provided

The terminal displays “hello, running Singularity with the file provided” as same in `hello.txt`.

Command –

```
podman ps -a
```

It shows container list

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
28ec6196ff06	localhost/sample_app:latest	in/sh -c cat /a...	19 seconds ago	Exited (0) 18 seconds ago		amazing_matsumot

---

Singularity (Apptainer)

Recipe --

Singularity employs a different method, using a "**recipe**" file that defines the steps to build an image in HPC. The "singularity build" command is then used to create the container image from the specified recipe.

Example:

1. Create a recipe (.def file) and an `app/input` folder containing the hello.txt “hello, running Singularity with the file provided” in the current same directory. def file -- use text editor to create it and save it as .def.

Recipe: hello.def

```
From: singularityhub/ubuntu

%runscript
export PATH=/usr/bin:$PATH
cat /app/input/hello.txt

%post
mkdir /code

%files
app/input/hello.txt /app/input/hello.txt
```

---

app/input folder:

(Same content as in previous examples)
It contains `hello.txt` file for running with the command “cat /app/input/hello.txt” as described in the Dockerfile.

2. Build image:
Command –

```
singularity build hello.sif hello.def
```

```
80027256@red2 lolcow]$ singularity build hello.sif hello.def
INFO:    Starting build...
INFO:    Use cached image
INFO:    Copying hello.txt to /input/hello.txt
INFO:    Running post scriptlet
+ mkdir /code
INFO:    Adding runscript
INFO:    Creating SIF file...
INFO:    Build complete: hello.sif
```

The Image “hello.sif ” is built.

3. Run the container:
Command –

```
singularity run -B $(pwd):/app/input hello.sif
```

```
80027256@red2 lolcow]$ singularity run -B $(pwd):/app/input hello.sif
hello, running singualrity with file provided
```

The terminal displays “hello, running Singularity with the file provided” as same in `hello.txt`.

Running Singularity using a Container Created by Podman

On cluster RED (HPC root), use Apptainer (Singularity) to run containers and Docker (Podman) for building.

Note: Running Apptainer needs to be on HPC root, not on a compute node.

Storing the Container Image

Previously, we had a container from the Podman image.

Check image list:
Command –

```
podman images
```

```
---------------------------------------------------------------------------------------------------------------------------
REPOSITORY                                TAG         IMAGE ID      CREATED         SIZE
localhost/sample_app                      latest      0c1e914e51db  8 seconds ago   80.4 MB
---------------------------------------------------------------------------------------------------------------------------
```

Check container list:
Command –

```
podman ps -a
```

```
-------------------------------------------------------------------------------------------------------------------------------------------------------
CONTAINER ID  IMAGE                           COMMAND           CREATED         STATUS                     PORTS       NAMES
28ec6196ff06  localhost/sample_app:latest     in/sh -c cat /a...  19 seconds ago  Exited (0) 18 seconds ago              amazing_matsumot
-------------------------------------------------------------------------------------------------------------------------------------------------------
```

On RED (HPC root), a convenience script called "mof-docker-upload" can save the Docker container to permanent network storage.

Command –

```
mof-docker-upload localhost/sample_app
```

80027256@red2 podman_example]$ mof-docker-upload localhost/sample_app
Uploading 5436af1495b6 to localhost:5000/nemp-students-and-trainees/localhost/sample_app ...

The script executes **4 Docker commands** as following for you. Once they have all executed successfully, your image will be uploaded to **localhost:5000** and permanent storage.

---

executed these commands for you:

docker commit --author 28ec6196ff06 nemp-students-and-trainees/sample_app
docker tag nemp-students-and-trainees/sample_app localhost:5000/nemp-students-and-trainees/sample_app
docker push --tls-verify=false localhost:5000/nemp-students-and-trainees/sample_app
docker image rm localhost/nemp-students-and-trainees/sample_app

Your image has been uploaded to **localhost:5000** and permanent storage.

The container 28ec6196ff06 and its image will be deleted from this machine in 7 days

To **run docker image** sample_app please execute the first command
below on a login node and the second one inside a Slurm batch script on any node.

apptainer pull --no-https docker://localhost:5000/nemp-students-and-trainees/sample_app

apptainer exec sample_app_latest.sif [command] [script-or-datafile]

Running the Container Interactively

Command (pull) –

apptainer pull --no-https docker://localhost:5000/nemp-students-and-trainees/sample_app

---

80027256@red2 podman_example]$ apptainer pull --no-https docker://localhost:5000/nemp-students-and-trainees/sample_app
INFO: Using cached SIF image

This command downloads the Docker image and converts it into an Apptainer image called `**sample_app_latest.sif**` in the current directory.

Command (run) –

apptainer **run** sample_app_latest.sif

80027256@red2 podman_example]$ apptainer run sample_app_latest.sif
hello, running singualrity with file provided

---

or

apptainer **exec** sample_app_latest.sif **cat /app/input/hello.txt**

80027256@red2 podman_example]$ apptainer exec sample_app_latest.sif cat /app/input/hello.txt
hello, running singualrity with file provided

The terminal displays "hello, running Singularity with the file provided" as same in `hello.txt`.

Running the Container in Batch Mode

We can execute the container by running `sbatch` with the `--wrap` option or by putting the Apptainer command in a shell script.

Command –

```
sbatch --output=output.txt --wrap "apptainer exec sample_app_latest.sif cat /app/input/hello.txt"
```

80027256@red2 podman_example]$ sbatch --output=output.txt --wrap "apptainer exec sample_app_latest.sif cat /app/input/hello.txt"
Submitted batch job 7698700

A file output.txt is created in current directory.

References

https://docs.sylabs.io/guides/3.5/user-guide/introduction.html

http://wiki.hpc.moffitt.org/HPC/software/containers/