Advanced Docker Course

Advanced Docker Course

- Dockerizing an application
- Deploying with Docker

Images and Containers

- Container: lightweight virtual machine
- Image: snapshot of a container

Simple commands

To start a new container

```
docker run -d -t ubuntu:18.04
```

finds the container ID

```
docker ps
```

runs a command in the container

```
docker exec suspicious_davinci ”ls” ”-lpF”
```

Goals

- using docker to deploy an application
- also in HPC setting
- understand different use cases:
  1. single application, temporary container
  2. complex application, temporary container
  3. lightweight Virtual Machine, always-on container

Container - Reproducibility

- Reuse existing images with a precise version of OS and software
  - Docker facilitates this integrating the concept of “reuse if possible” in its core
- Use a **Dockerfile** file to describe all the steps of creating and configuring a container.
- Stateless: data are connected but are not part of the application
  - Docker has Volumes to “contenerize data”
  - also bound directories

Architecture

Docker architecture

Image — Container

Image is static, immutable

Container is dynamic, mutable

1 container $\rightarrow$ 1 image

1 image $\rightarrow$ several containers

docker run

```
docker run -d -t ubuntu:18.04
```

run: command

-d: option detached

-t: create a terminal

ubuntu: image

18.04: image version (label)

docker run

```
docker run -i -t ubuntu:18.04 /bin/bash
```

run: command

-i: option interactive

-t: attached to your terminal

ubuntu: image

18.04: image version (label)

/bin/bash: program to run inside the container

When the program completes the execution, the container is stopped (but not deleted)

docker run

```
docker run --rm hello-world
```

– **rm**: deletes the container (not the image)

docker run: name

```
docker run -it --name vm ubuntu:18.04 /bin/bash
```

-it: is like `-i -t`

–name: name of the *container*

Sometimes a random name is not a good idea

docker run: name

```
docker run -it --rm --pid=host  ubuntu:18.04 top
```

–**pid=host**: the container and the host computer share the same process IDs (just like a native process)

docker run: limiting resources

```
docker run -dt --name vm -m 1g ubuntu:18.04 /bin/bash
```

-m: max amount of memory available

–memory-swap: max amount of swap memory. If not specified is equal to the max memmory (in this case 1GB)

–cpus: number of cpus available. Can be a fraction

docker run: command arguments

```
docker run -t  -m 1g ubuntu:18.04 ls -lph /var
```

Just place them after the command.

docker run: environment

```
docker run -it -e ”DEBUG=true” ubuntu:18.04 ls -lph /
```

Set the value of an environment variable.

A separate **-e** for each variable.

docker exec

```
docker exec vm df -h
```

Runs a command (in this case **df -h**) in a running container

docker images

```
docker images
```

List the images

docker image inspect

```
docker image inspect hello-world
```

Gives several information on the image

docker rmi

```
docker rmi hello-world
```

Deletes a local image. It has no effect on the hub.

docker ps

```
docker ps
```

List the running containers

```
docker ps -a
```

List all containers

docker rm

```
docker rm vm
```

Deletes a container and all its data (volumes)

docker system prune

```
docker system prune
```

Reclaim all disk space

docker system df

```
docker system df
```

Shows what is using disk space