Getting started with containers

Managing containers and images

Two important concepts

Image ⬅ Container

- read-only
- stored on longer term
- can be used as a base

- based on the image
- short-lived
- usually only minor adjustments

The concept of layers

- In docker: images have a starting point
- User makes changes/installations
- Stored in a layer on top of existing
- Creating an image:
  - From a container: `docker commit` (not reproducible)
  - From a `Dockerfile`

The concept of layers

container
image layer - add pysam
image layer - add python
base image - Ubuntu
Kernel

container: writeable

image: read-only
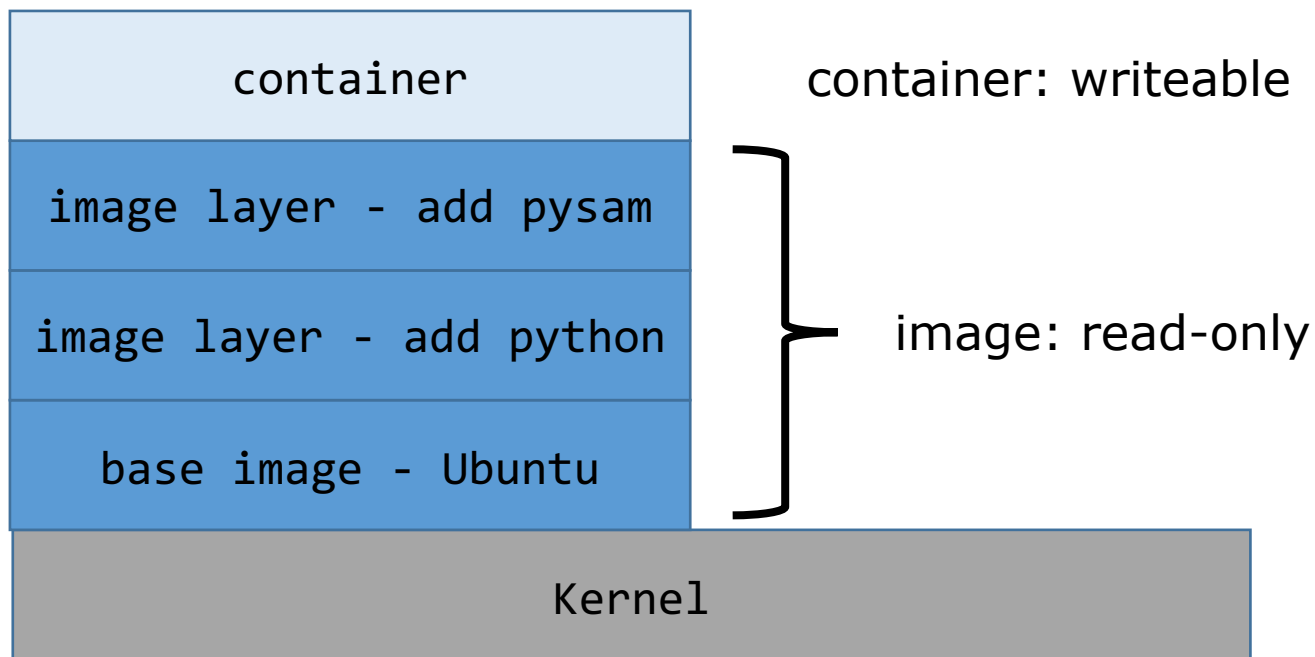
Quiz question 3

The docker engine

- Manages both the
    - images
    - containers
- Layers are efficiently handled
- Images and containers will remain available unless specified
- Two important commands:
    - `docker image ls`
    - `docker container ls`

Sharing an image

- docker hub (open to the world)
  - Alternatives: quay.io, singularityhub, gitlab and github container repositories …
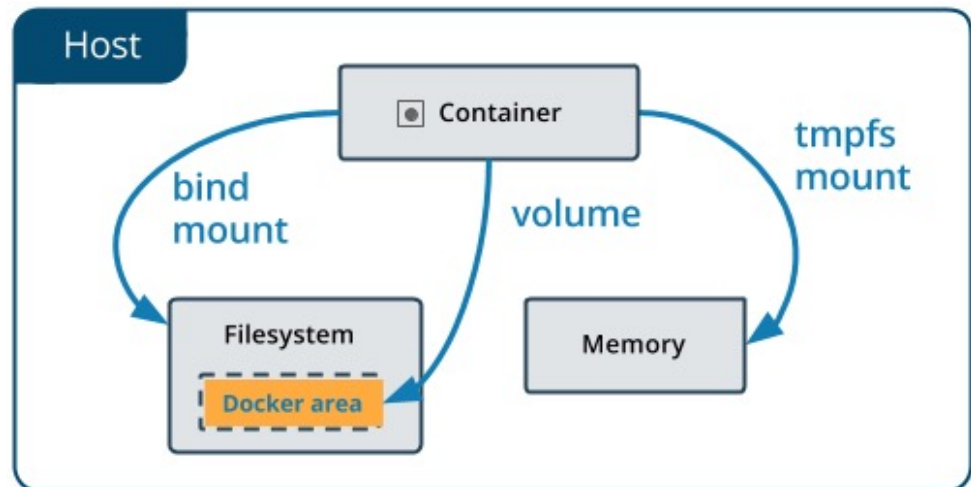- command docker save
- Dockerfile

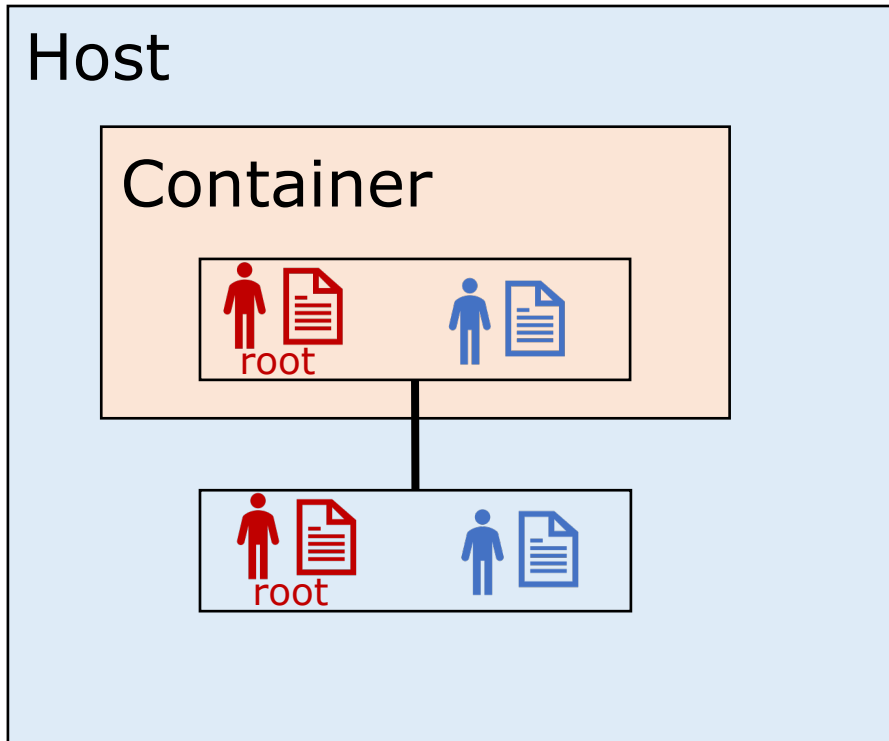# Question 4

Frequently used features

- Mounting directories
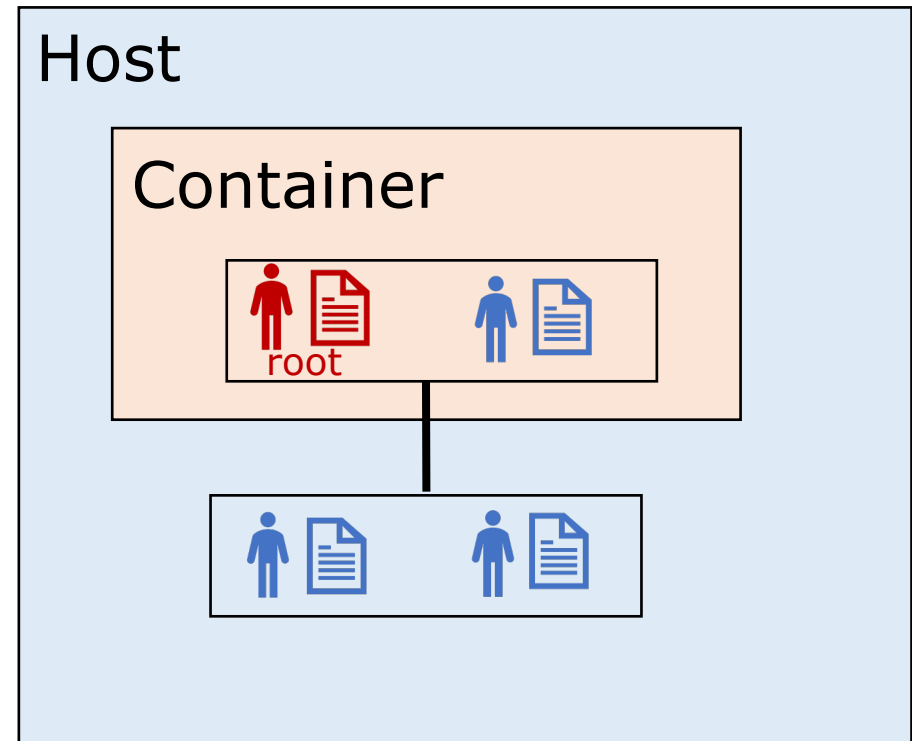- Managing identities
- Mapping ports

Mounting

- **Bind-mount:** Make a directory on the host available to the container
- **Volume:** Disk space reserved and managed by docker (isolated)

Identity

Linux

Other systems

```
docker run -u "$(id -u):$(id -g)"
```

Mapping ports

- Processes that display browser content:
  - Jupyter
  - Rstudio server
  - Any other web server
- These are published at [IP]:[PORT], so e.g: 127.0.0.1:8000
- Forward the port from the container to port on the host: `docker run –p 80:8000`
- Meaning: publish port 8000 in the container at port 80 on the host