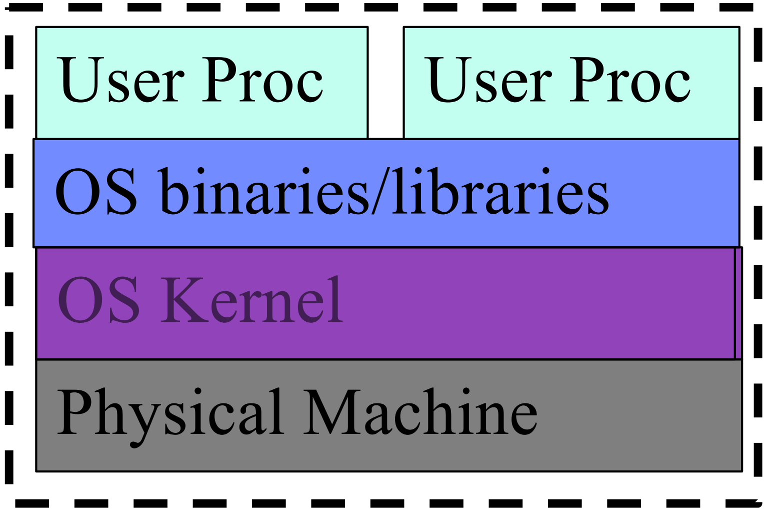
THE UNIVERSITY OF WESTERN AUSTRALIA

Virtualization: UniApps (Linux Lab) vs Docker

Lecture 2

Daniel Smith
Michael J Wise

Single Operating System

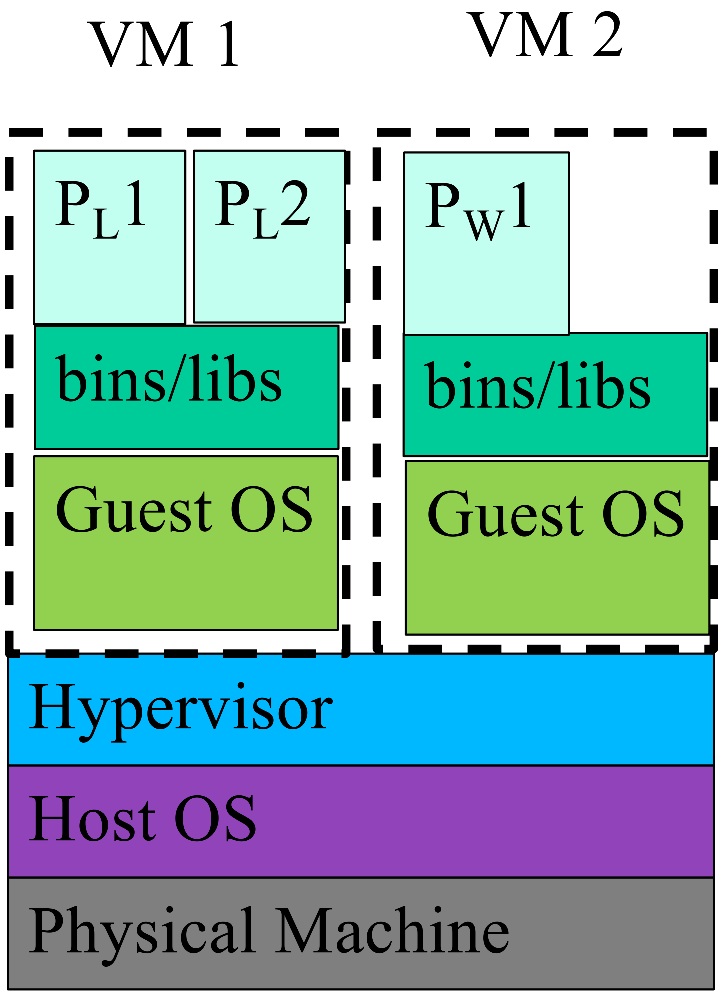


- Conventional system architecture, e.g. Unix running on a laptop or large, shared machine
- Each process has own space, but most things shared
- Each user has own disk space
- Can only run apps for that architecture
  - *Emulation possible, e.g. Wine. Can be slow*

Virtual Machines

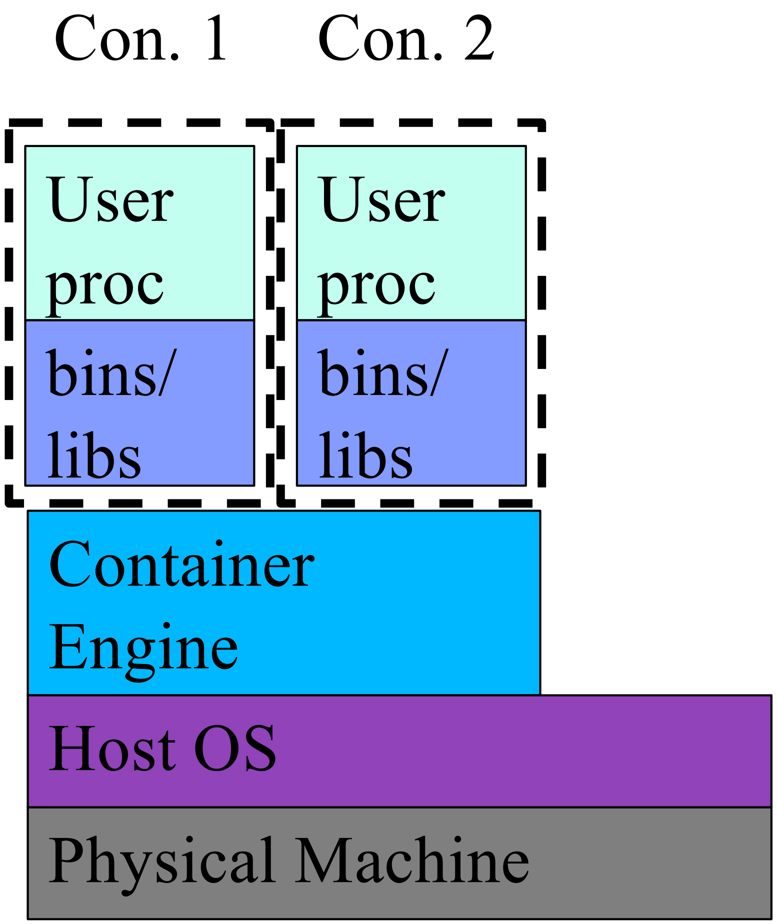
- VM stacks entire OSs next to each other, including each one's kernel.
- Coordination by *hypervisor*
- *Each VM isolated from the others*
- Once "spun up" a VM can support multiple users
- Provides consistent environments
- Centrally maintained

Virtual Machine (UniApps)



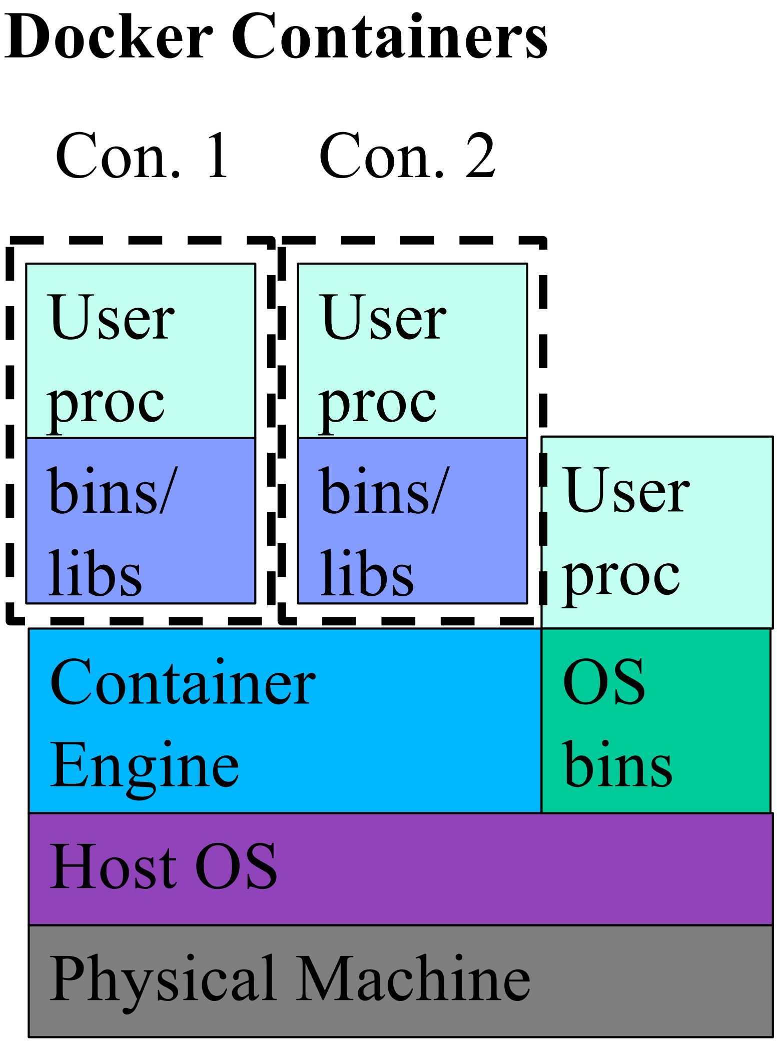
Containers - Docker

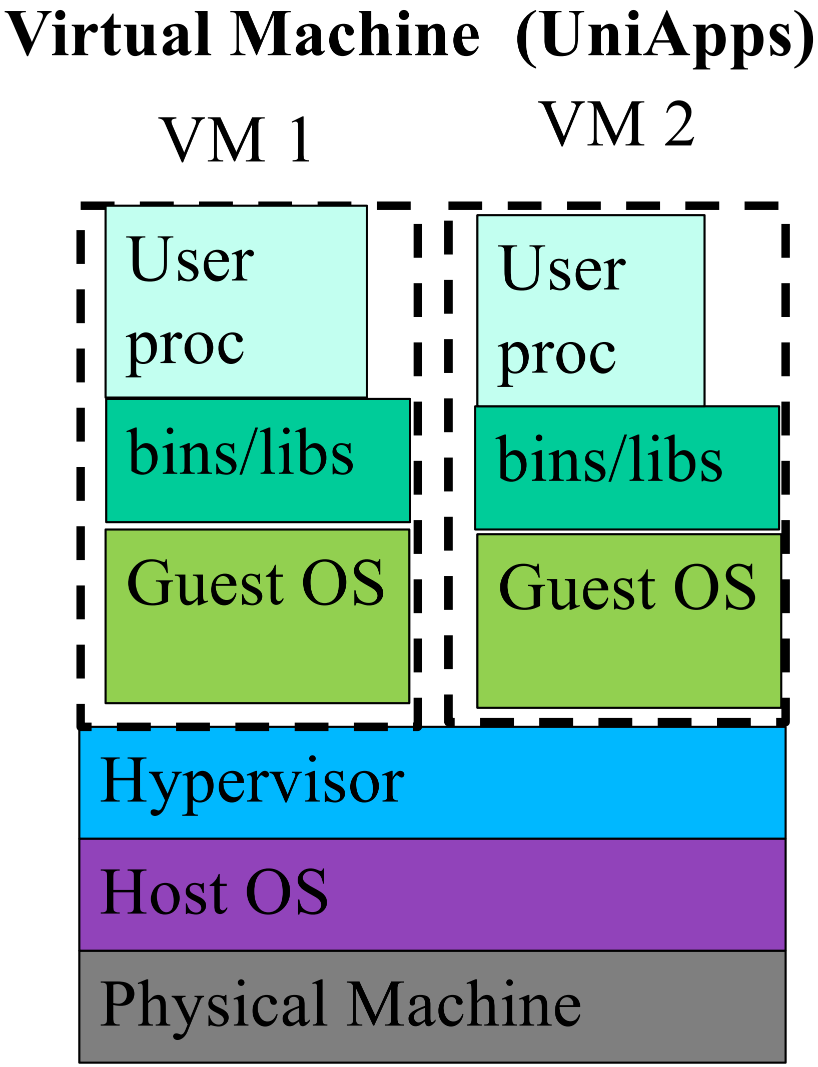
Docker Containers



- Containers contain application code plus as much of the OS support as required, e.g. libraries, binaries, called *dependencies*
- Container engine, e.g. Docker, is link to kernel and lower levels of host OS.
- Consistent experience for an app. Different between apps
- Distributed maintenance; User initiates updates

Docker Containers vs Virtual Machine





Containers vs Virtual Machine

Conventional architecture

- + efficient
- – cannot support apps from other Operating Systems or older version of the same OS

Containers vs Virtual Machine

Virtual Machines (e.g. UniApps)

- + A VM can support multiple users (like a conventional architecture); each VM isolated so security enhanced. Can run any app by mounting compatible system; Centralized control
- – Carries the cost of each OS, binaries, kernel, etc. so large footprint on disk and in memory; performance may suffer; costlier to update centrally

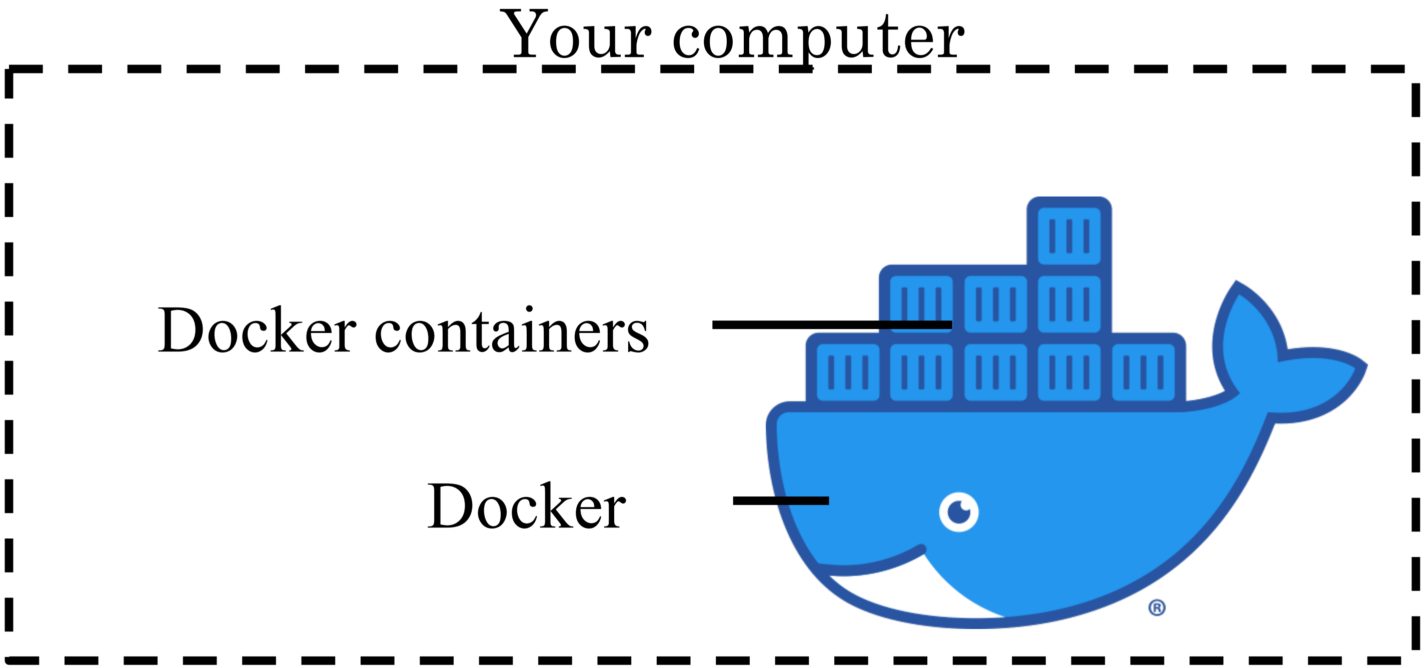
Docker Containers vs Virtual Machine

Containers (e.g. Docker)

- + Only carry around as much of the environment as they require. App larger than for native OS, but portable; Use host kernel, smaller footprint which is distributed to users
- – Weaker security than VM

Docker

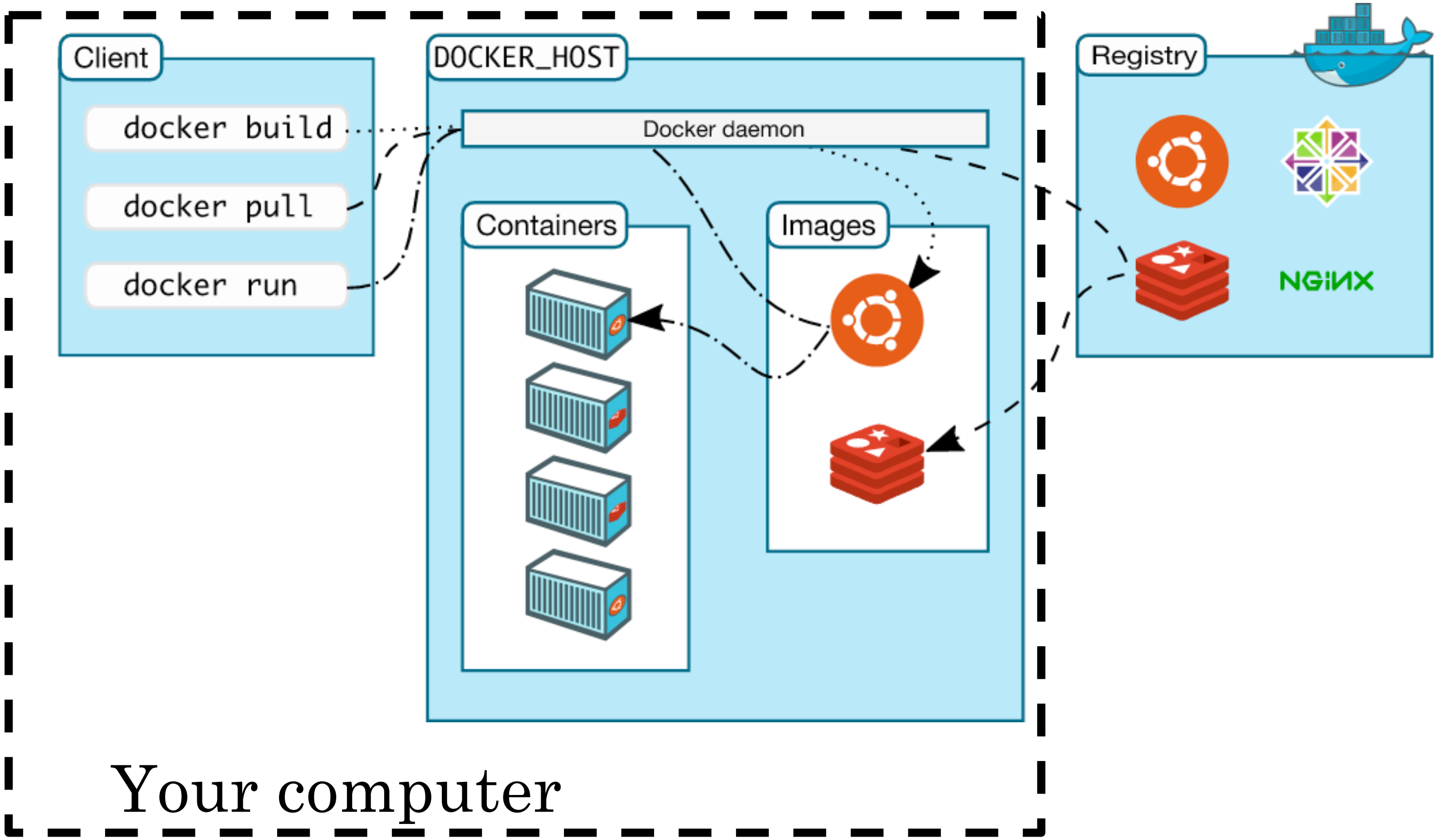
- Docker is one of a number of systems that implement containers
- https://docs.docker.com/desktop/mac/install/
- https://docs.docker.com/desktop/windows/install/



Docker

- Docker containers are typically **stateless** (container storage is erased when the container stops)!
  - *This includes temporary files, so need link to file system outside container for results, etc.*
- Online Docker registries allow straightforward deployment of Docker images
- Docker is free for educational use

Docker Architecture



Source: https://docs.docker.com/get-started/overview/

Docker Architecture

- **Client:** The way you interact with Docker. For us, this will be on the command line.
- **Daemon:** A background service that listens for API requests and manages Docker objects.
- **Docker Desktop:** An easy-to-install application that includes a client (both CLI and GUI), a daemon, and other stuff.
- **Docker registry:** A place where Docker images are stored. We will use Docker Hub, a public registry that is configured in Docker by default.
- **Docker Objects:**
  - *An **image** is a read-only template for a container.*
  - *A **container** is a runnable instance of an image that can be started/stopped/paused etc.*

Running Docker

- Assume you have done: `mkdir cits4407`
- Launch docker with mounted directory:

```
docker run \
--mount type=bind,source=/usr/me/cits4407,\
target=/home/stud/perm \
-it mjw263/cits4407_2024:v1
```

The full path to cits4407

Container to be loaded

`--mount` takes a directory from your real computer and links it to the container at `/home/stud/perm`.

`-it` Runs the named container as an interactive terminal

- Leave Docker and stop the container with `exit`

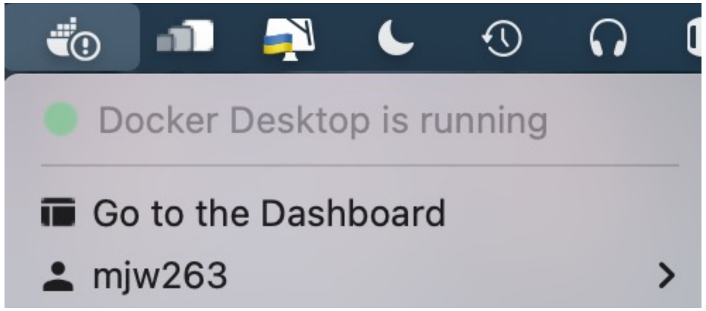
WARNING

- Remember, Docker containers are stateless. This means that **when a container stops, everything inside it will be deleted forever**.

- To avoid losing your work, **make sure you save it in your mounted directory**
  - *In this case* `/home/stud/perm` so that you will later find it at `~/cits4407`

Things to Lookout For (aka Gotchas!)

- Make sure the daemon is running



Docker Desktop is running

- Run docker with mounted directory using

  `--mount`
- Use absolute paths with `--mount`
- Launch containers from the command line, not Docker Desktop

Further reading

https://docs.docker.com/get-started/

https://docs.docker.com/engine/reference/run/

https://docs.docker.com/storage/bind-mounts/