RESEARCH TECHNOLOGIES
UNIVERSITY INFORMATION TECHNOLOGY SERVICES

CORNELL UNIVERSITY
FOUNDED A.D. 1865

Center for Advanced Computing

RESEARCH TECHNOLOGIES
UNIVERSITY INFORMATION TECHNOLOGY SERVICES

Deep Dive into Constructing Containers for Scientific Computing and Gateways: Introduction to Containers

Eric Coulter, Jeremy Fischer, Stephen Bird, Sanjana Sudarshan, Peter Vaillancourt and Suresh Marru

UITS Research Technologies, Indiana University &
Center for Advanced Computing, Cornell University

Escience 2021 – Sept 20, 2021

Outline for the day's events…

Part I

- Introduction to container technologies (You are here!)
- Gateways Tie-In
- Environment Introduction
- Simple Docker Container Creation
- Complex Docker Container Build
- First exercise – building and running a scientific workload with Docker and Singularity

Part II

- Singularity Containers + Exercise
- Docker to Singularity Conversion
  - Best Practices
  - Sample Conversion
  - New Docker Features

Outline for the day’s events…

Part III

- Open Q&A
- Workflow Exercises
- WRF Demo
- Next Steps and Best Practices

Part VI

- Containers for Gateways + Exercises
- Closing Q&A

Introduction

…or how I learned to stop worrying and love ~~the bomb~~ containers…

What are containers?

- Born from a simple idea (chroot)
- Evolved over time into various forms of container services (e.g. BSD Jails, Solaris Zones)
- LXC (LinuX Containers) was released in 2008
- Docker came on the scene in 2013
- Other technologies evolved – Shifter, CharlieCloud, Singularity
- Upping the ante – Docker Compose and container orchestration

…and why would I want to use one

- Consistency
- Portability
- Ability to package and run on HPC
- “Just in time” instantiation and updating on the fly
- Creating microservices
- Run legacy code/obsolete OSes
- Reproducible Science!

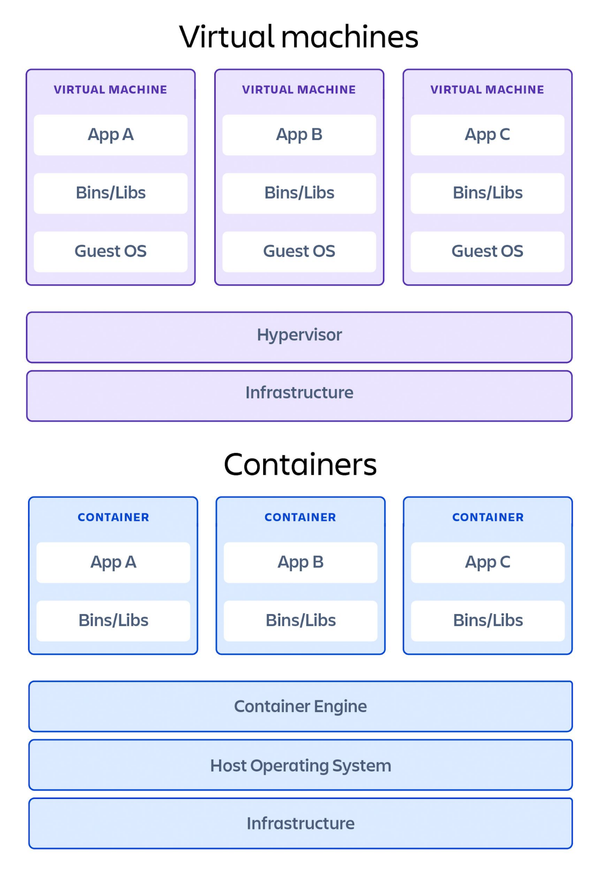
Containers vs. Virtual Machines

VMs:

- VMs are fully contained – everything you need is there
- VMs are independent of the host operating system
- All OS resources and tools are available

Containers:

- Compact – minimal OS parts to run, rely on host
- Compact nature makes them more portable
- Robust ecosystem – many pre-made containers available



Virtual machines

VIRTUAL MACHINE
App A
Bins/Libs
Guest OS

VIRTUAL MACHINE
App B
Bins/Libs
Guest OS

VIRTUAL MACHINE
App C
Bins/Libs
Guest OS

Hypervisor

Infrastructure

Containers

CONTAINER
App A
Bins/Libs

CONTAINER
App B
Bins/Libs

CONTAINER
App C
Bins/Libs

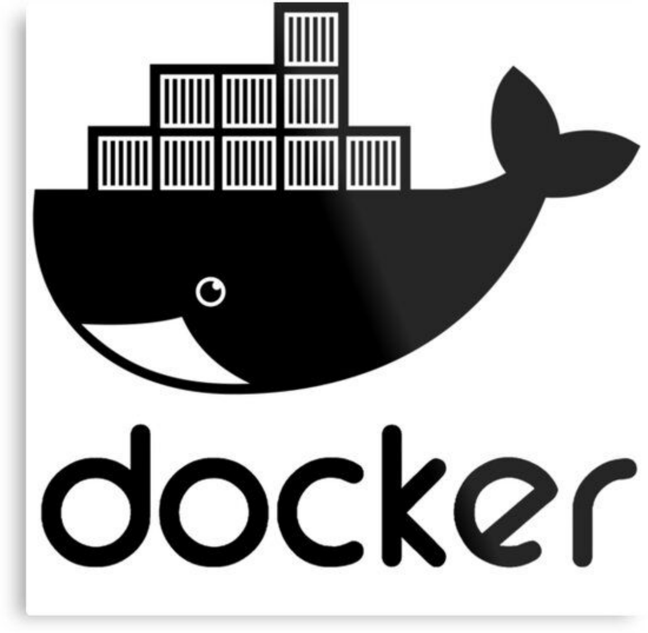
Container Engine

Host Operating System

Infrastructure

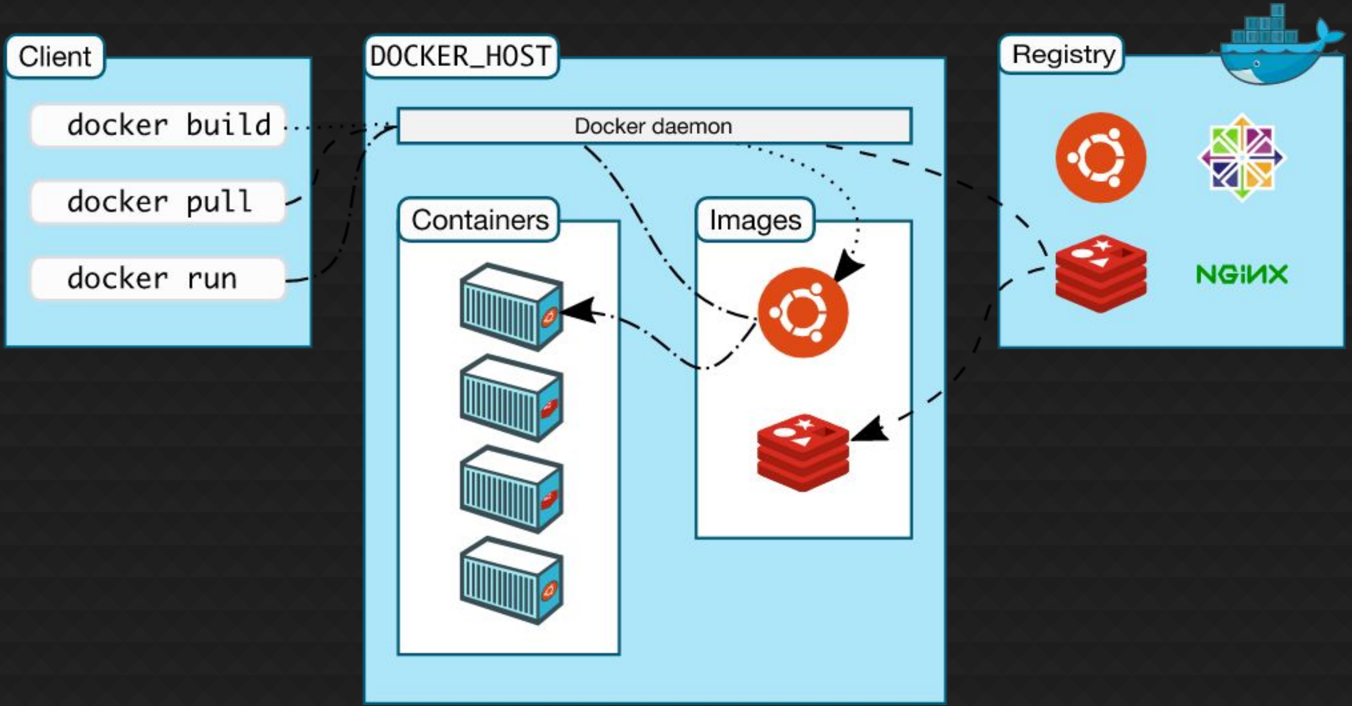
What is Docker

- Docker is a container technology tool to create, deploy, and run applications
- Low overhead, uses the running kernel
- Lets a creator package all of the software needed to run an application in a reasonably compact and run it on any other Docker-capable machine**
- Uses a client (*docker)* to talk over a REST API to the docker daemon (*dockerd)* either locally or remotely
- Has a large public repository of objects (containers, images, etc) at DockerHub -- and other repos are available
- It allows users to develop applications, package (ship) them into containers which can then be deployed anywhere

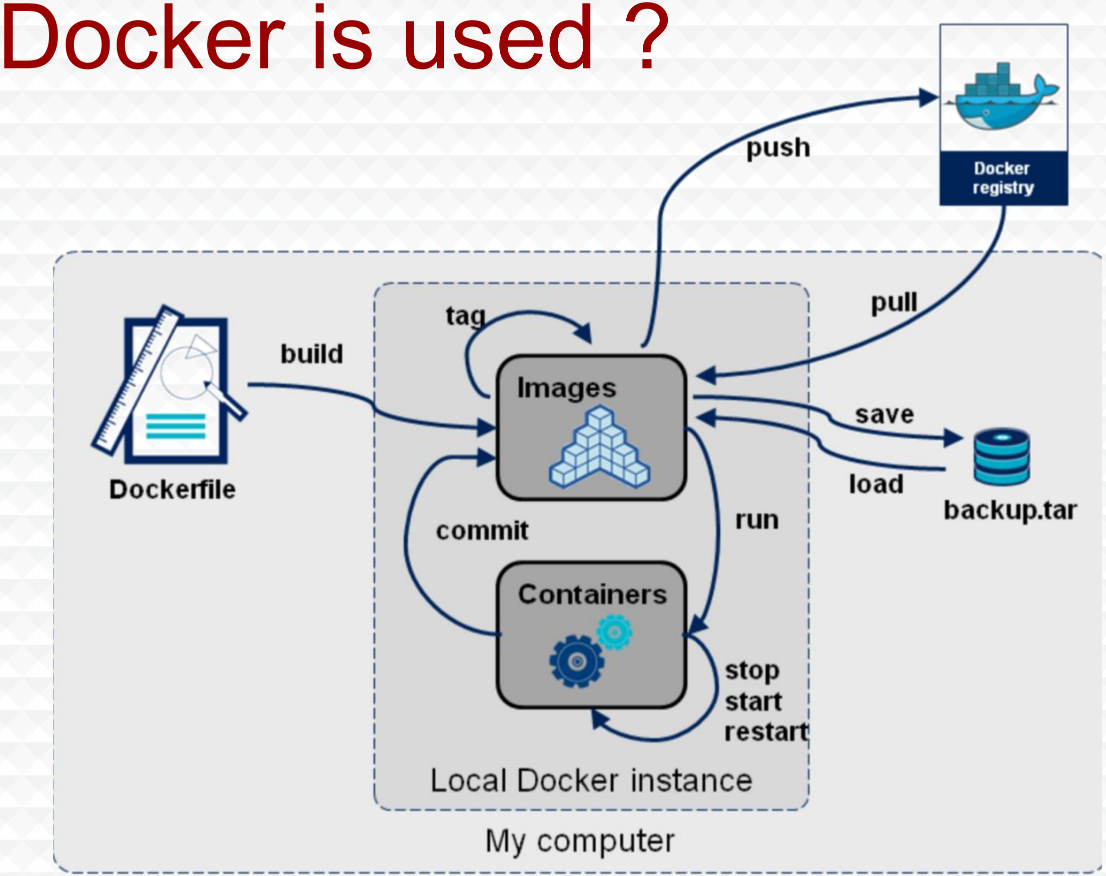


** Even Windows and Mac!

Docker in action…the big picture:



How Docker is used ?



What is Singularity?

- YACP (Yet another container platform)
- Why are we talking about Singularity at all?
- How is it different from Docker?
- How does this all come together?
- Singularity can use Singularity containers from a Singularity-compatible hub or Docker containers, even pulling from a Docker registry like Docker Hub

Moving into the first exercise…

Simple container creation!

Questions?