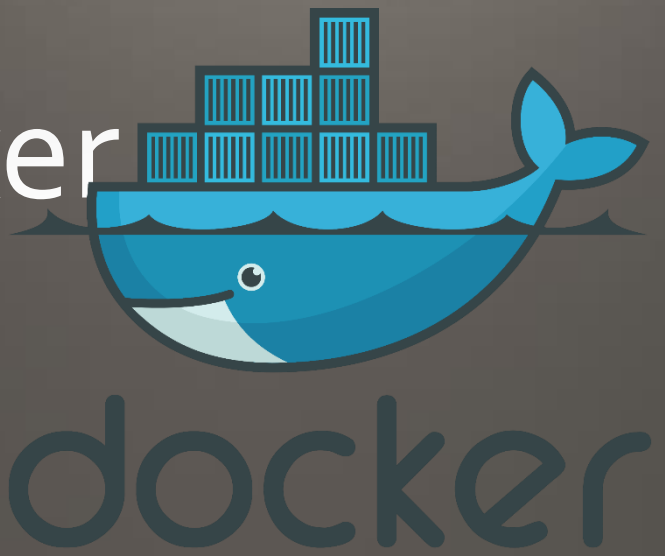
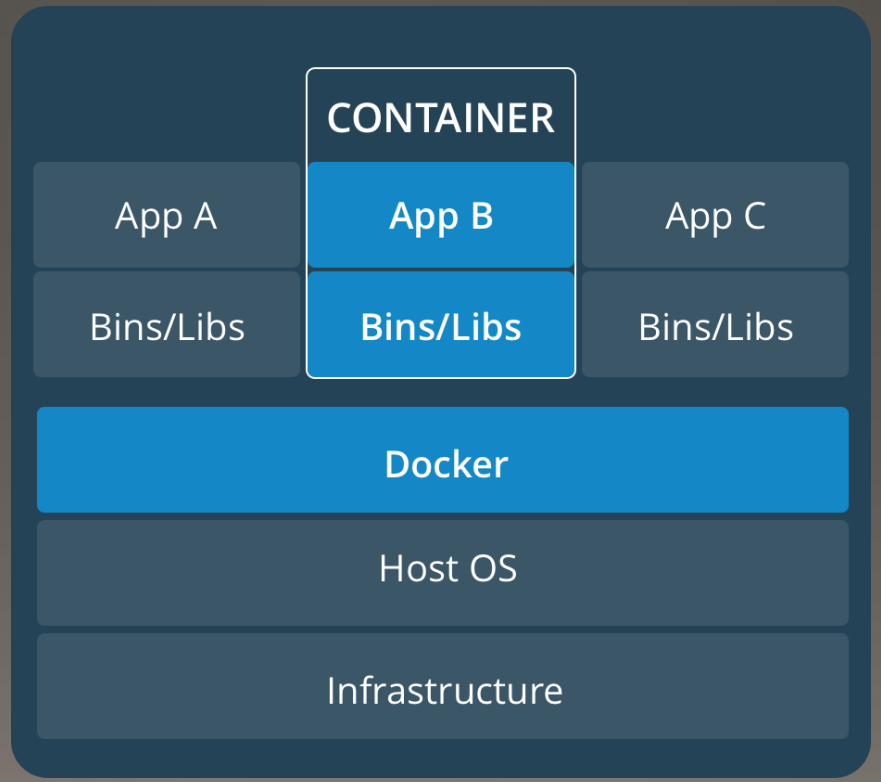
Jacob Boschee

Docker in VMs

Containers and Docker



- What is a container?
  - Software boxes
  - Linux chroot
- What does Docker provide?
  - Full bundle of software
  - Static, reproducible environments
  - Hardware and OS agnostic



Why use Docker for my research?

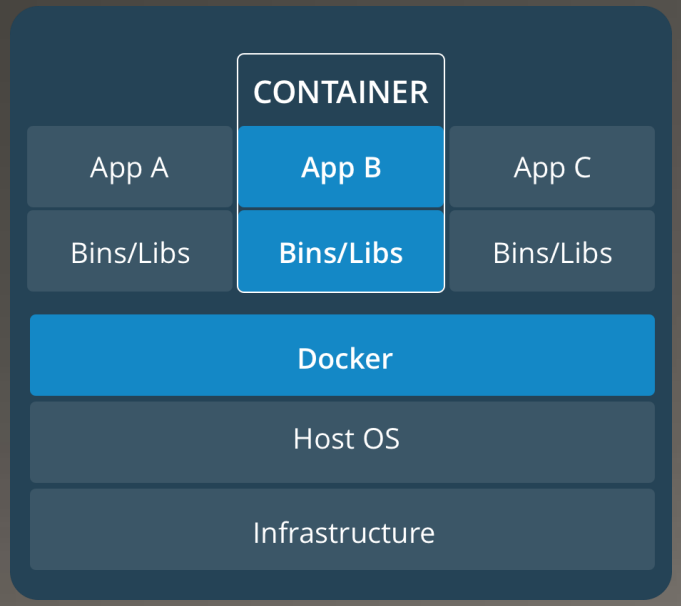
- Reproducibility
  - Need to make sure your results can be duplicated
  - Software dependency trees can be massive and making sure all packages match is challenging

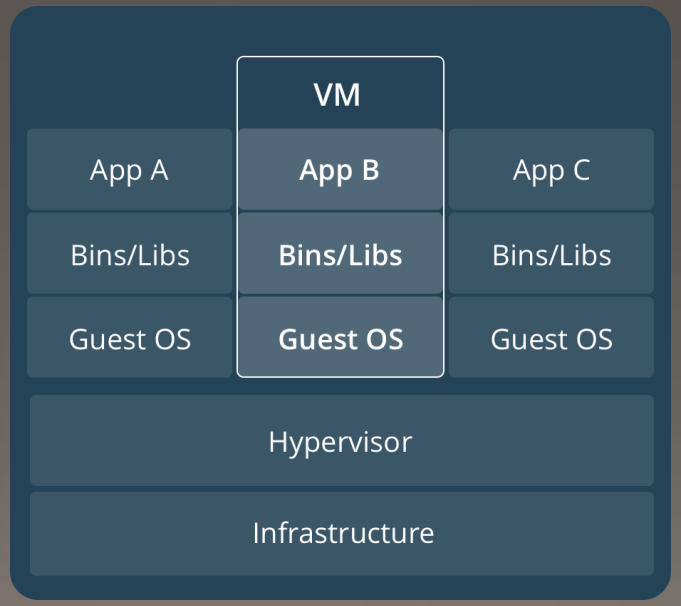
Why user Docker for my Research?

- Software environments
  - Does your research depend on software that has been out of development for years (or even decades)?
  - Do you need access to libraries that are no longer offered by current system distributions?
  - How many hours does it take to set up your environment for your software when you need to deploy to a new system?

Docker structure vs VM structure

- Docker
  - Stateless
  - Run within a common Linux kernel
  - Accesses hardware through the kernel interfaces
- VM
  - Statefull
  - Run multiple kernels under a hypervisor
  - Each kernel accesses hardware through the hypervisor





Dockers and Repositories

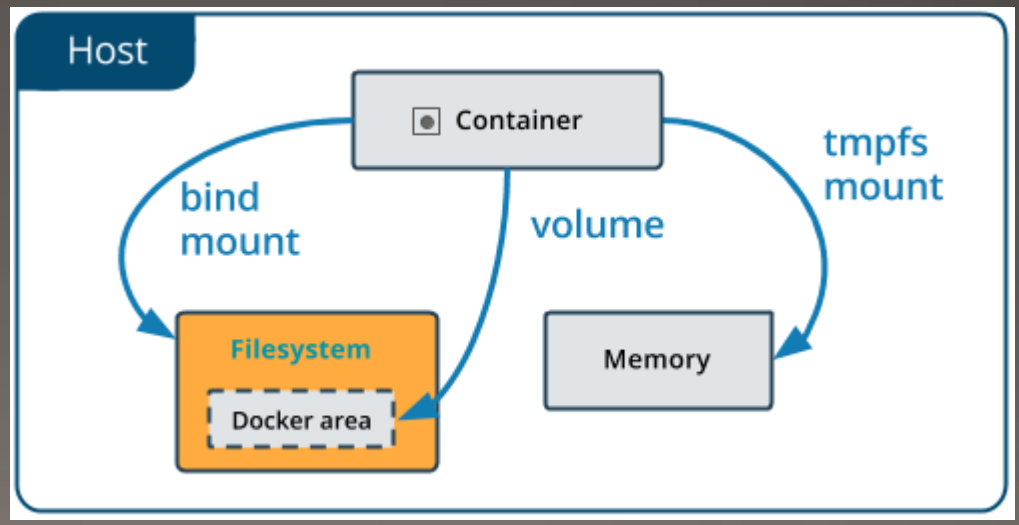
- Docker Hub
  - Base clean images to build new custom Docker containers from
- Github
  - Specialized Docker images for scientific workflows
- Local custom built images for your workgroup

Installing and setting up your Docker

- See Exercise #1 and Exercise #2

Interfacing with Docker containers

- Launching a shell within a Docker container
  - docker run and docker start with the -i flag
- Stateless but still flexible
  - Docker environment variables
  - Docker external mounts
  - Docker forwarded ports



Docker Environment Variables

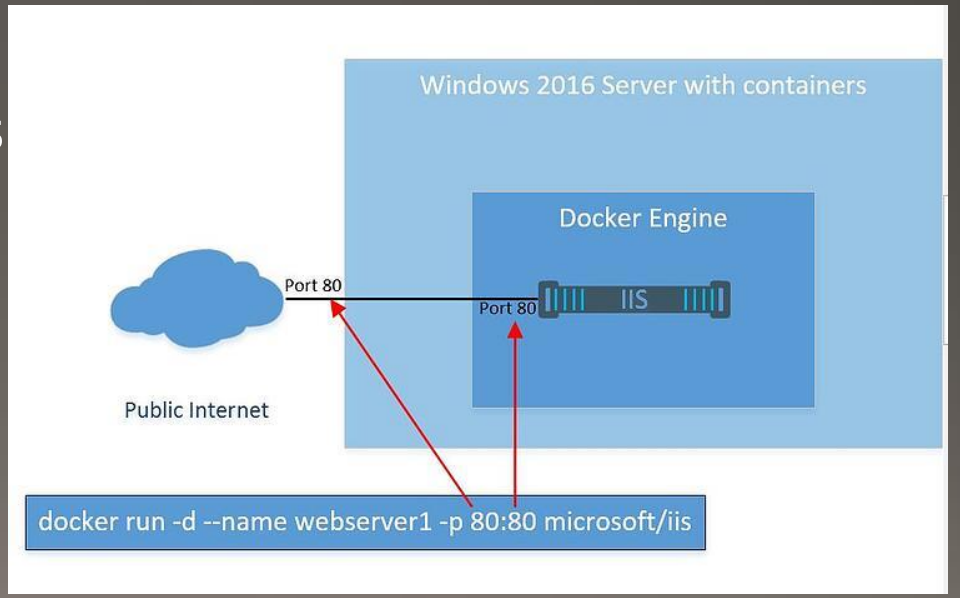
- External configuration of Docker containers at runtime
- Environment variables can be used to pass input parameters
  - File names
  - Software calculation settings
  - External web locations for input files
- Use the -e flag to define on runtime
  - docker run -e OMP_THREADS=32 docker_image

Docker external mounts

- Docker maintains its own private filesystem within its container
- Mounting from the host system is possible however
- Allows full directories to be shared with the host and container for ease of importing files or exporting results
- Use -v flag to define mount points from the host to the container
  - docker run -v /tmp/my_inputs:/tmp/my_workdir docker_image
  - Files in or written to the directories on the host or container side are preserved on the host even when removing containers

Docker forwarded ports

- Some Docker containers need access to external ports
  - Jupyter Notebooks (8888)
  - Apache web connections (80 and 443)
- They can be mapped from the host of the container into the container by use of the -p flag
  - docker run -p 8888:8888 docker_image
  - The host will forward incoming connections to the first specified port to inside the container with the second specified port



Docker Execution and Shell

- See Exercise 3:

Images and Containers

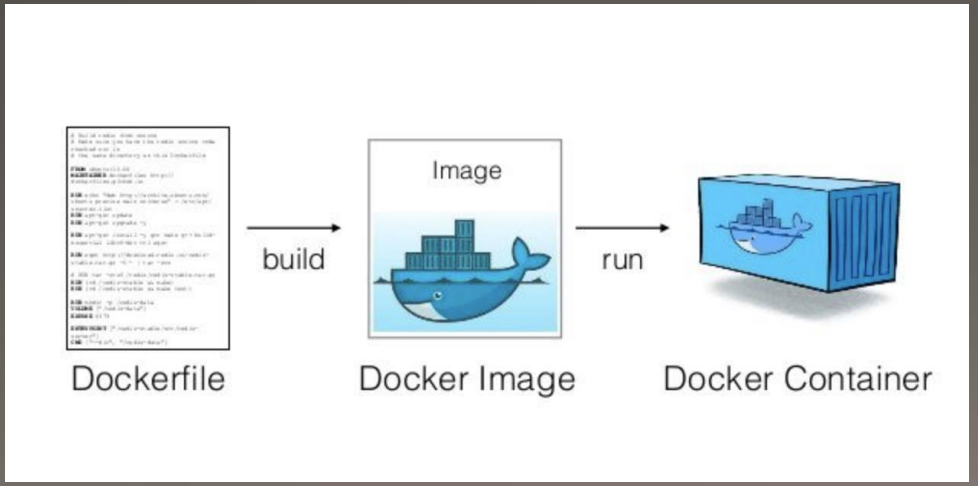
- Immutable vs Stateful
  - Once built images will always produce the same initial state when used to create a new container
  - Containers will remember their state between starting and stopping
- Best practice is to create an image with an initial setup that can run the container to completion and then delete the container
  - This ensures reproducibility with the workflow

Modifying Containers

- Interactive sessions
  - Grants a user a full root shell within the container
- Modifications made will persist within the container until it is removed
- Image that the container was created from does not change

Building New Images

- Multiple way to get new images for Docker
  - DockerHub for standardized images
  - Importing Docker images generated by other users
  - Using a current container and 'commit'ing the container to create an image
  - Using a Dockerfile to build a new image off of an existing image



Docker Files

- Allows a user to specify and build a new Docker image to their needs
- Runs a set of commands on building the image to prepare the state
- For running non-interactively sets a command and work directory for the image
- May also set default Docker options that can be overridden at run time
  - Mounts, Environment Variables, Ports, ect.

Building a Custom Image

- See Exercise 4

Singularity and HPC Containers

- Docker and elevated permissions
  - On multi-user systems this will be restricted
- Singularity is the solution for HPC
- Excellent way to port complicated python environments
- ComputeCanada's guide to Singularity: https://docs.computecanada.ca/wiki/Singularity

Question and Answers