Docker & Containers for Science

Joe Zuntz

Overview

Talk

- Motivations
- Containers
- Docker & Dockerfiles
- Data storage
- Scientific Usage
- Case Study
- HPC
- Image Management

Tutorial

- Installing
- Running a container
- Dockerfiles
- Lifetimes
- Data
- Jupyter
- Advanced Dockerfiles
- Docker Hub
- Exploring Images

Motivations

- Portable & repeatable environments
- Easy installation of complete software stacks
- Virtual machines too heavy
- Easy mass deployment
- Application isolation

Containers

- An approach to solving these problems
  - with bundled up file systems
  - and isolated processes

Ways to think of containers

- Packaged up file system
- Lightweight virtual machine
- Application with bundled dependencies
- Sandboxed system

Container Frameworks

- Docker
- CoreOS rkt
- Garden (Cloud Foundry)
- LXD

- Ecosystem complicated!
- Make crowding but significant interoperability

- Market-leading container framework
- Open source framework and open container format
    - Images easily exported to other frameworks
- Well documented and supported

★Tutorial 1: Installation★

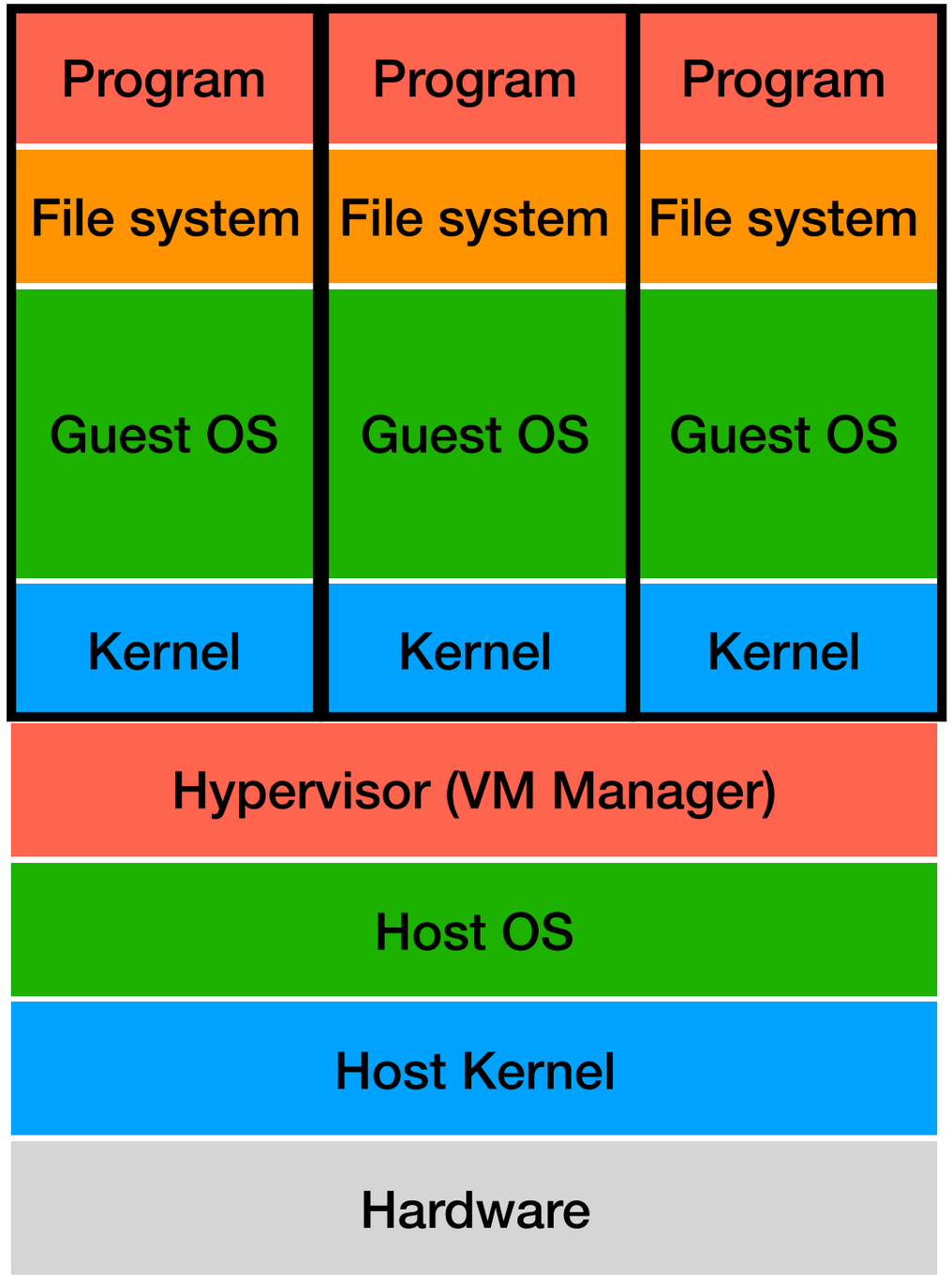
- Mac: https://store.docker.com/editions/community/docker-ce-desktop-mac
- Windows: https://store.docker.com/editions/community/docker-ce-desktop-windows
- Linux: https://docs.docker.com/engine/installation/
    - Go to “Server” and find your OS.
    - Select “CE” (community edition) if it gives you two choices
    - May also be in your package manager (yum/apt)
- `docker pull ubuntu`

Glossary

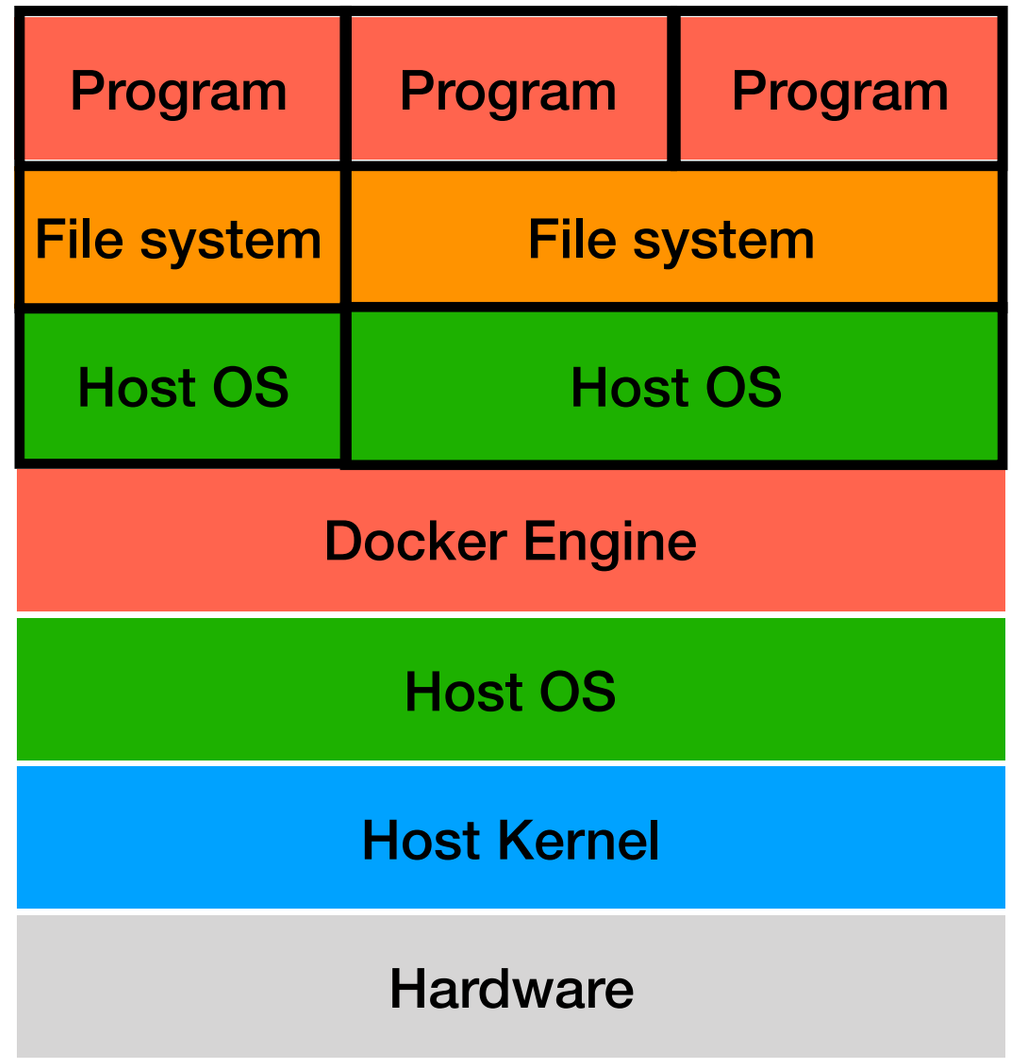
Term	Meaning
Docker	Framework for building and sharing images and running containers
Image	Frozen snapshot of a file system
Container	Instance of an image with processes running in it
Dockerfile	Recipe for making images
DockerHub	Shared repository of images

Compared to Virtual Machines

- Dramatically faster startup
    - Can package single command as container start
- Much lighter
- Less isolated from host machine



Virtual Machine



Docker
(under Linux)

Mac & Windows

- Docker “really” only runs under Linux
- On Mac & Windows tiny linux layer run as VM
- No CPU performance impact
- Sometimes disk I/O impact
- Issues with permissions means it actually works better on a mac than linux

Tutorial 2: Running

- `docker run --rm -it ubuntu bash`
- Run some commands and explore the system, create some files, download some packages, etc.
- In another tab: `docker ps`
- `exit` when complete
- `docker images`

Tutorial 2: Running

- `docker run --rm -it ubuntu bash`

- --rm means the container is deleted after the command stops
    - Data you create in the container is deleted with rm.
    - Otherwise have to delete manually later
    - Alternative: use a single long-running container
- -i and -t make the container properly interactive

Dockerfiles

- Recipe for building images
- Sequence of commands
- Steps cached so quick to modify
- Usually start from existing image
- Will discuss in detail during tutorial

```
FROM ubuntu:16.10
MAINTAINER joezuntz@googlemail.com
#Joe's note to himself.  Compile this with: docker build -t joezuntz/cosmosis-base
#then docker push joezuntz/cosmosis-base

# Basic compilers and tools dependencies
RUN apt-get update -y && apt-get install -y gcc g++ gfortran wget make python-dev \
    pkg-config curl \
    && apt-get clean all

# Manual installation of mpich seems to be required to work on NERSC
RUN mkdir /opt/mpich && cd /opt/mpich \
    && wget http://www.mpich.org/static/downloads/3.2/mpich-3.2.tar.gz \
    && tar xvzf mpich-3.2.tar.gz &&  cd mpich-3.2 && ./configure && make -j4 \
    && make install && rm -rf /opt/mpich

# The environment variables needed by the CosmoSIS build and runtime.
ENV GSL_INC /usr/include
ENV GSL_LIB /usr/lib/x86_64-linux-gnu
ENV CFITSIO_INC /usr/include
ENV CFITSIO_LIB /usr/lib/x86_64-linux-gnu
ENV FFTW_LIBRARY /usr/lib/x86_64-linux-gnu
ENV FFTW_INC_DIR /usr/include
ENV MINUIT2_LIB /usr/local/lib
ENV MINUIT2_INC /usr/local/include

# Run a bash login shell if no other command is specified.
CMD ["/bin/bash", "-l"]
```

Tutorial 3: Dockerfiles

- Create an empty directory and a file in it called Dockerfile:

```
FROM  ubuntu:latest
LABEL maintainer="your_email@example.com"
RUN  apt-get update && apt-get install -y python3 python3-pip
```

- From that directory run

```
docker build -t my-image .
```

★Tutorial 3: Dockerfiles★

- Add these line to end of your Dockerfile:

```
RUN pip3 install jupyter ipython numpy
```

- Build again:

```
docker build -t my-image .
```

- Run a container:

```
docker run -it --rm  my-jupyter  ipython
```

★Tutorial 3: Dockerfiles★

- Lines executed sequentially
- Each line creates a new layer in file system
- -t identifies image with tag

Docker for Scientists

- We mostly run programs not services
- Environments can be fiddly and custom
- More likely to be CPU and disk limited than network
- Our users are often developers too
- Use clusters and supercomputers in locked-down environments

Docker Target Market

- Web servers and related components
- Enterprise services
- Long-running processes like databases
- Cloud computing

- Not aimed at scientists =>
  Weird behaviour & docs

Tutorial 4: Lifetimes

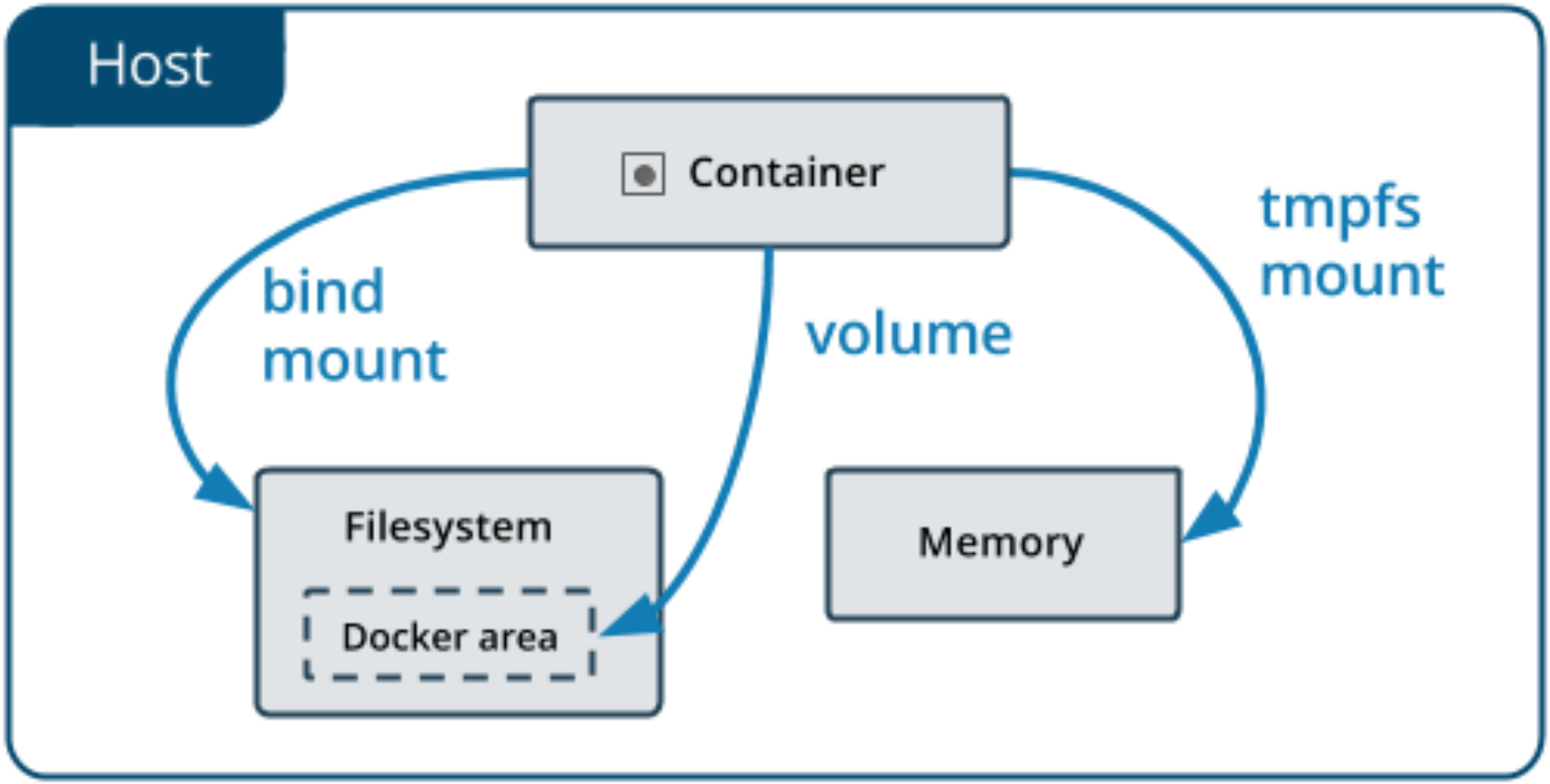
- **`docker run -it ubuntu bash`**
- Create some large files and then exit
  e.g. cat > my.txt
- **`docker ps -as`**
- Container is stopped but not removed:
- **`docker start -i`** ***`container_number`***
- Can only re-run the same command you started with
- Task: Figure out how to use the **`docker rm`** command to remove your container

Tutorial 4: Lifetimes

- While other container is running, new terminal:
- **`docker exec -it container_number bash`**
- Run a new command in a running container

Data Storage

- By default container is isolated from your disk
- Can ad-hoc copy things from containers
- Three Options for more automatic solutions
    - Temporary storage — “tmpfs”
    - Docker-managed storage — “volumes”
    - Mounted host directories — “bind mounts”


Host
Container
bind
mount
volume
tmpfs
mount
Filesystem
Docker area
Memory

Tutorial 5: Data

- `docker run -it --rm  my-jupyter bash`

- `ipython`

```
import numpy as np
x=np.random.randn(1000)
np.savetxt("data.txt", x)
```

- Task: figure out how to use the `docker cp` command to get data.txt out

Tutorial 5: Data Bind mounts

- Make a new directory you want to share:
- `mkdir ./data`
- `docker run -it --rm -v $PWD/data:/data my-jupyter bash`
- `/data` now mounted inside image
- Can use this in clever ways!

Tutorial 5: Data
Bind mounts

- `docker run -it --rm -v $PWD/data:/data my-jupyter bash`
- Full path is always required for both elements.
- Can have multiple -v commands for more mounts

Science Use Cases

- Distributing complex programs & frameworks
    - e.g. large project analysis suites
- Versioning analysis pipelines
- Running Databases
- Continuous Integration
- Cloud computing?

Case Study: CosmoSIS

- CosmoSIS project - cosmological parameter estimation
- Many dependencies, multi-language architecture
    - GCC, GSL, CFITSIO, FFTW, Lapack, Git, Python 2.7, Numpy, Scipy, Nose, PyYaml, Matplotlib, Emcee
- Large majority of issues associated with installation
    - Consistency very hard to maintain for manual installs

Case Study: Issues Page

#254: Make fails	installation
#251: Installation error on Ubuntu 16.04	installation
#250: Error with Sierra installation using the bootstrap ups branch	installation
#249: Compilation error related to multinest & libgfortran	installation
#248: make fails on first installation	installation
#247: problems with manual install	installation
#242: Installation error in docker method: outdated pip?	installation
#234: Error with manual installation	installation
#217: Installation fail due to python 3.5 being installed	installation
#208: unable to run cosmosis example/example_a.ini on docker	installation
#207: Installation error OSX 10.9.3 : ImportError: No module named urllib3	installation
#200: el Capitan woes (linking libraries and seg fault)	installation
#194: About the setup script_setup-my-cosmosis	installation
#193: `GFORTRAN_1.4' not found	installation
#183: Unable to run make in RedHat 6 (initial buildup)	installation
#179: Cannot install matplotlib	installation
#177: Can't compile cosmosis	installation
#167: compiling error /usr/bin/ld: cannot find -lblas	installation
#166: Seg fault when running demo 16	installation
#161: Installation failing - URL for package no longer works	installation

Case Study: Requirements

- Users will modify code
- Users are busy and unused to docker
- Need to use host machine programs to edit code and view results
- Program are CPU-intensive but I/O-light
- Some components (e.g. Lapack) sensitive to CPU architectures

Case Study: Docker Approach

- Build Docker image packaging most dependencies
- Distribute Dockerfile to compile CPU-sensitive components locally
    - Builds upon global image we distribute
    - Users can add their own pieces easily
- Mount code directories from host machine
    - Can be edited from host
    - Outputs saved directly to host machine
- Provide simple scripts to build and launch

Case Study: User Interface

Reduced many hours of from-scratch user installation to two commands:

```
./get-cosmosis-and-vm ./cosmosis
./start-cosmosis-vm ./cosmosis
```

Does not affect any other installation method

Communication

- By default containers can connect out to world, access internet etc.
  But outside cannot connect inwards.
- Must expose ports with -p flag if want to connect inwards
    - e.g. to run databases, servers, notebooks

Tutorial 6: Jupyter

- `docker run --rm -it -p 8888 my-juypter jupyter notebook --port=8888  --ip=0.0.0.0 --allow-root`

- Open the URL in your browser - need to change 0.0.0.0 t0 127.0.0.1

- Task: Combine these commands with the data mounting we looked at earlier to save notebook and results to host disk.

Tutorial 6: Jupyter

- **`docker run --rm -it -p 8888 my-juypter`** ***`jupyter notebook --port=8888  --ip=0.0.0.0 --allow-root`***
- Green italics = command run inside container
    - Could have just put “bash” then typed all this

Docker for HPC

- Pure docker generally unsuitable for HPC
    - Requires root
- Other solutions being developed in HPC sector
    - Singularity (LBL)
    - Shifter (NERSC)
- A little immature but moving fairly quickly

Singularity

- Lawrence Berkeley Lab container implementation
- No isolation from old file system
- Can build new images (with root access)
- Runs existing docker images (without root)
- Moderately well developed

Shifter

- NERSC container implementation
- No isolation from old file system
- Environment passed through
- Cannot build new images, just run existing ones
- Runs docker images
- Works but no clear error messages yet

Tutorial 7: More Dockerfile Directives

- CMD
- ENV
- COPY
- USER
- Task: Investigate these directives in the Dockerfile documentation
- Task: Use the CMD directive to make the running your notebook easier

Image Management

- Reason Docker is popular:
  hub.docker.com
- “pull” command goes there and finds image layers
- One private image by default
  - Pay for more and other services

★Tutorial 8: Docker Hub★

- Make an account at hub.docker.com
- `docker build -t username/my-jupyter .`
- `docker push username/my-jupyter`
- Pull your neighbour's image

Tutorial 9: Exploring Images

- Save your image to disk:
  docker save -o my-jupyter.tar my-jupyter

- Extract the tar file:
  `tar -xf my-jupyter.tar`

- Have a look around

- These layers are read-only but can be built on

    - Union File System