



# Containers

QLS 612 - May 2024  
Sebastian Urchs - Alyssa Dai



Fondation  
Brain Canada  
Foundation



HEALTHY BRAINS  
HEALTHY LIVES



Montreal Neurological  
Institute-Hospital



# Containers?



# What we will talk about

1. **Why** are containers useful for researchers
2. Using and building containers with **Docker**
3. Using containers on supercomputers with **Singularity**

# Why isolate environments

# Document your software environment



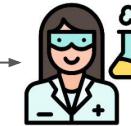
1: to share with colleagues



# Document your software environment



## 1: to share with colleagues



This repository contains code for running the alignment for cognition project. It builds on code from the recent preprint:

An empirical evaluation of functional alignment using inter-subject decoding.  
Thomas Buzziell<sup>1</sup>, Elizabeth DuPuis<sup>1</sup>, Jean-Baptiste Poline<sup>1</sup>, & Bertrand Thirion.  
doi: 10.1101/2020.12.07.211500

## Requirements

Dependencies :

- `fmalign`
- `ribabel>=3.1`
- `humpy>=1.18`
- `multidimlist`
- `gradio`
- `scipy`
- `lockfile`

## Installation

First, make sure you have installed all the dependencies listed above. Then you can install cog-align by running the following command:

```
cog-align
```

## 🔗 Requirements

## Dependencies

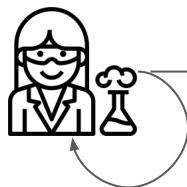
- fmralign
  - nibabel>=3.1
  - numpy>=1.18
  - matplotlib
  - pandas
  - scipy
  - scikit-learn

## Installation

First, make sure you have installed all the dependencies listed above. Then you can install cog-align by running the following commands:

```
git clone https://github.com/neurodatascience/cog-align  
cd cog-align  
pip install -e .
```

# Document your software environment



1: to share with colleagues



2: for yourself 3 months (days?) from now!

## What's New In Python 3.0



Python 2.7.18

```
>>> print "hello"
hello
```

**Author:** Guido van Rossum

### Print Is A Function

The `print` statement has been replaced with a `print()` function, with keyword arguments to replace most of the special syntax of the old `print` statement ([PEP 3105](#)). Examples:

```
Old: print "The answer is", 2*2
New: print("The answer is", 2*2)

Old: print x,          # Trailing comma suppresses newline
New: print(x, end=" ") # Appends a space instead of a newline

Old: print            # Prints a newline
New: print()          # You must call the function!
```



Python 3.11.7

```
>>> print "hello"
File "<stdin>", line 1
  print "hello"
  ^^^^^^^^^^^^^^
```

```
SyntaxError: Missing parentheses
in call to 'print'. Did you mean
print( ...)?
```



# Document your software environment



1: to share with colleagues



2: for yourself 3 months (days?) from now!

## What's New In Python 3.0



Python 2.7.18

```
>>> print "hello"
hello
```

**Author:** Guido van Rossum

### Print Is A Function

The `print` statement has been replaced with a `print()` function, with keyword arguments to replace most of the special syntax of the old `print` statement ([PEP 3105](#)). Examples:

```
Old: print "The answer is", 2*2
New: print("The answer is", 2*2)

Old: print x,          # Trailing comma suppresses newline
New: print(x, end=" ") # Appends a space instead of a newline

Old: print            # Prints a newline
New: print()           # You must call the function!
```



Python 3.11.7

```
>>> print "hello"
File "<stdin>", line 1
  print "hello"
  ^^^^^^^^^^^^^^
```

```
SyntaxError: Missing parentheses
in call to 'print'. Did you mean
print( ...)?
```

# Don't use the same environment for all projects



Changing dependencies may do unexpected things

- A. Your updated the dependencies of an existing project

# Do not install things into your system Python!

## Common installation issues

### Installing into the system Python on Linux

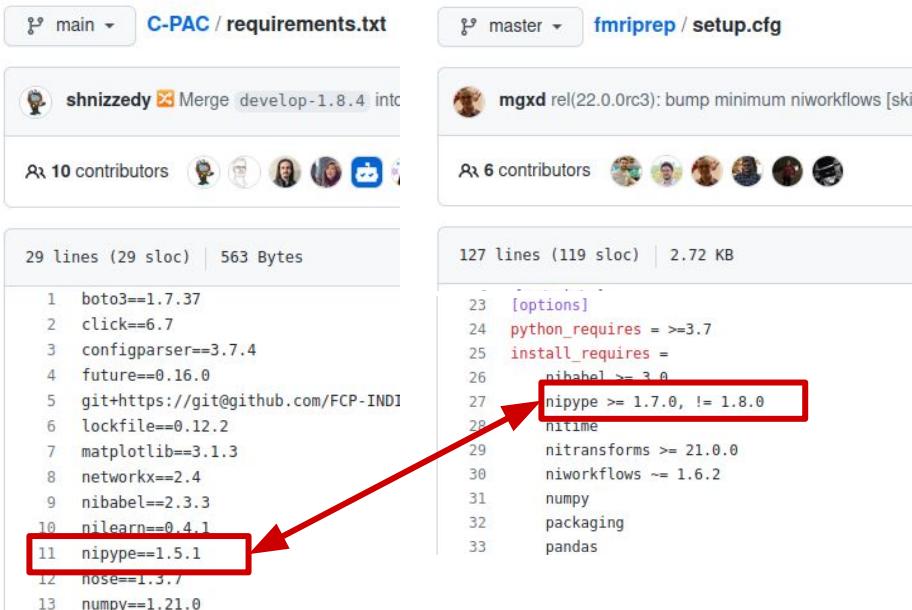
On Linux systems, a Python installation will typically be included as part of the distribution. Installing into this Python installation requires root access to the system, and may interfere with the operation of the system package manager and other components of the system if a component is unexpectedly upgraded using pip.

On such systems, it is often better to use a virtual environment or a per-user installation when installing packages with pip.

```
[surchs@marvin ~]$ which python  
/usr/bin/python  
[surchs@marvin ~]$ which pip  
/usr/bin/pip  
[surchs@marvin ~]$ ]
```

# Don't use the same environment for all projects

Changing dependencies may do unexpected things



The image shows two GitHub repository pages side-by-side. The left page is for 'C-PAC / requirements.txt' and the right page is for 'fmriprep / setup.cfg'. Both pages show the file content, commit history, contributors, and file statistics.

**C-PAC / requirements.txt**

shnizzedy Merge develop-1.8.4 into master · 11 commits · 10 contributors · 29 lines (29 sloc) | 563 Bytes

```
1 boto3==1.7.37
2 click==6.7
3 configparser==3.7.4
4 future==0.16.0
5 git+https://git@github.com/FCP-INDI
6 lockfile==0.12.2
7 matplotlib==3.1.3
8 networkx==2.4
9 nibabel==2.3.3
10 nilearn==0.4.1
11 nipype==1.5.1
12 nose==1.3.7
13 numpy==1.21.0
```

**fmriprep / setup.cfg**

mgxd rel(22.0.0rc3): bump minimum niworkflows [skip ci] · 1 commit · 6 contributors · 127 lines (119 sloc) | 2.72 KB

```
23 [options]
24 python_requires = >=3.7
25 install_requires =
26 nibabel >= 3.0
27 nipype >= 1.7.0, != 1.8.0
28 nitime
29 nittransforms >= 21.0.0
30 niworkflows ~ 1.6.2
31 numpy
32 packaging
33 pandas
```

A. Your updated the dependencies of an existing project

B. Two projects use the same environment but need different versions of some dependencies

# Consider: a cake

Home · Cakes · Perfect Cream Cheese Pound Cake

## Perfect Cream Cheese Pound Cake

Published by [Sally](#) on February 18, 2019 - [700 comments](#)



NOT SPONSORED, BUT I ABSOLUTELY ADORE NORDIC WARE BAKING PARTS.

10-12 cups of batter. [\*\*This one\*\*](#) is also gorgeous! 😊

- 5 **Bake:** Bake the cream cheese pound cake at 325°F (163°C). Half the cake with aluminum foil to prevent over-browning.
- 6 **Cool, then Invert:** Let the pound cool for about 2 hours in the plate and cool completely before serving.

# Consider: a cake

Home · Cakes · Perfect Cream Cheese Pound Cake

## Perfect Cream Cheese Pound Cake

Published by [Sally](#) on February 18, 2019 - [700 comments](#)



NOT SPONSORED, BUT I ABSOLUTELY ADORE NORDIC WARE BUNDT PANS.

10-12 cups of batter. [\*\*This one\*\*](#) is also gorgeous! 😊

- 5 **Bake:** Bake the cream cheese pound cake at 325°F (163°C). Half the cake with aluminum foil to prevent over-browning.
- 6 **Cool, then Invert:** Let the pound cool for about 2 hours in the plate and cool completely before serving.

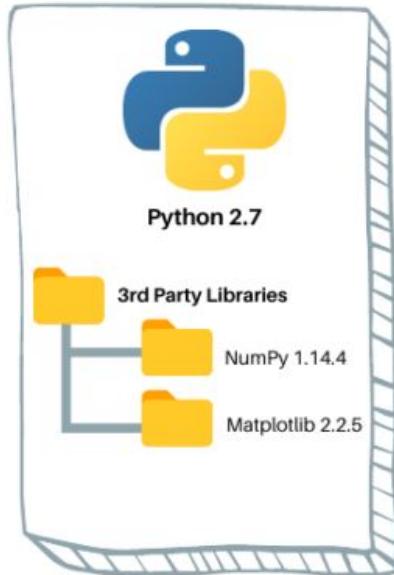


# isolate environments to handle different requirements

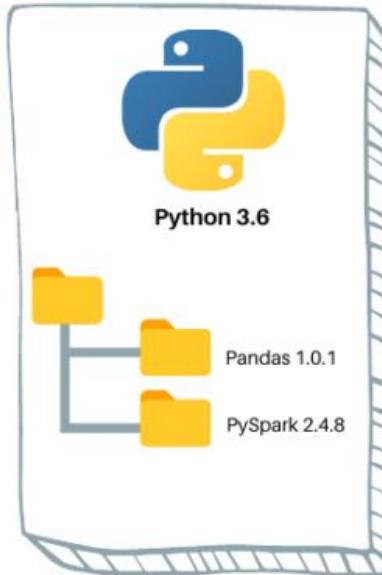


# isolate Python environments

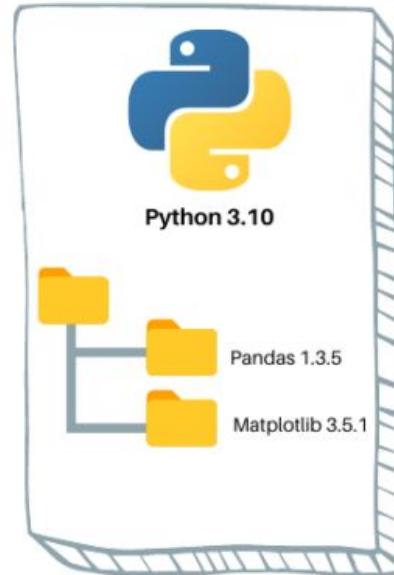
Virtual Environment 1



Virtual Environment 2



Virtual Environment 3



# Why not just python virtual environments?

## External Dependencies

fMRIprep is written using Python 3.8 (or above), and is based on [nipyne](#).

fMRIprep requires some other neuroimaging software tools  
that are not handled by the Python's packaging system (Pypi)  
used to deploy the [fmriprep](#) package:

- FSL (version 6.0.5.1)
- ANTs (version 2.3.3 - NeuroDocker build)
- AFNI (version 22.3.06)
- C3D (version 1.3.0)
- FreeSurfer (version 7.3.2)
- ICA-AROMA (version 0.4.5)
- bids-validator (version 1.8.0)
- connectome-workbench (version 1.5.0)

- Not all binary dependencies are on Anaconda
- Not everything is written in Python or R
- Your Operating System (**OS**) also has packages and a package manager
- The same version problems apply to these

## Need to capture the (entire) compute environment





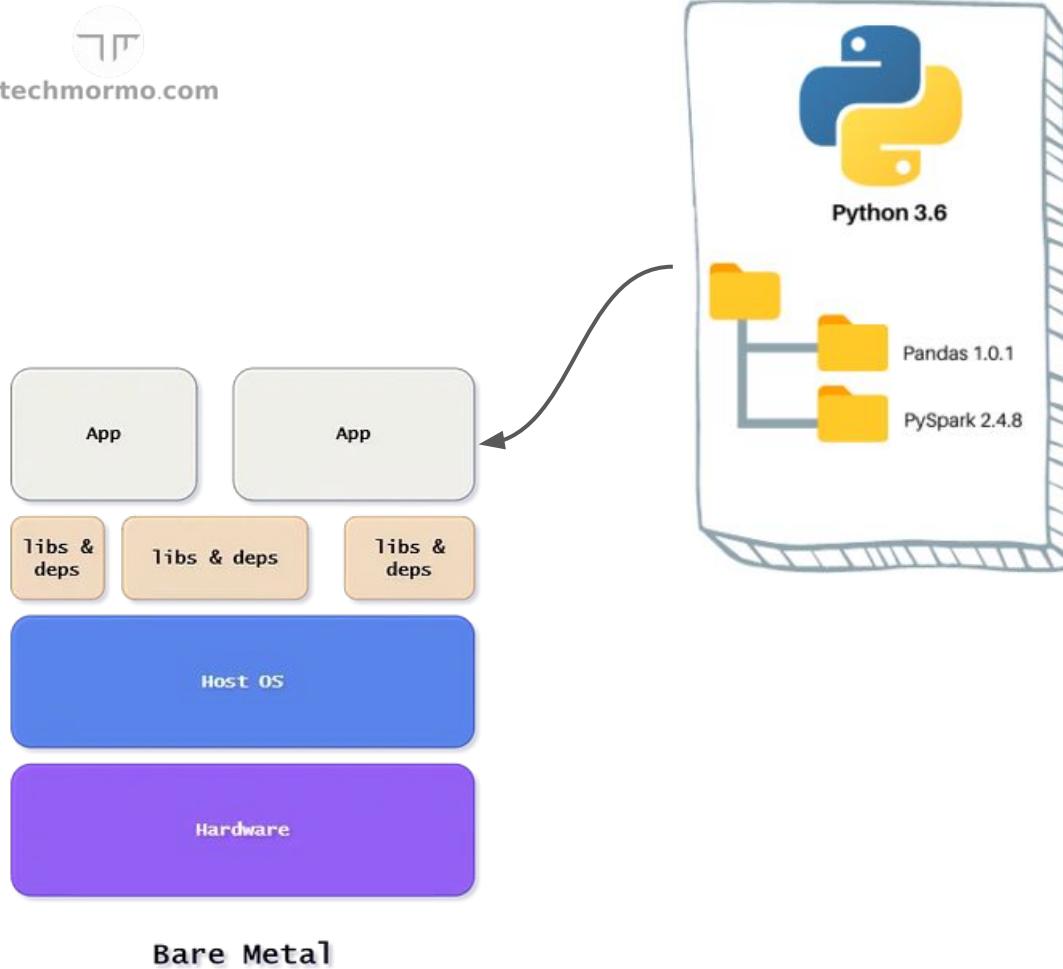
techmormo.com

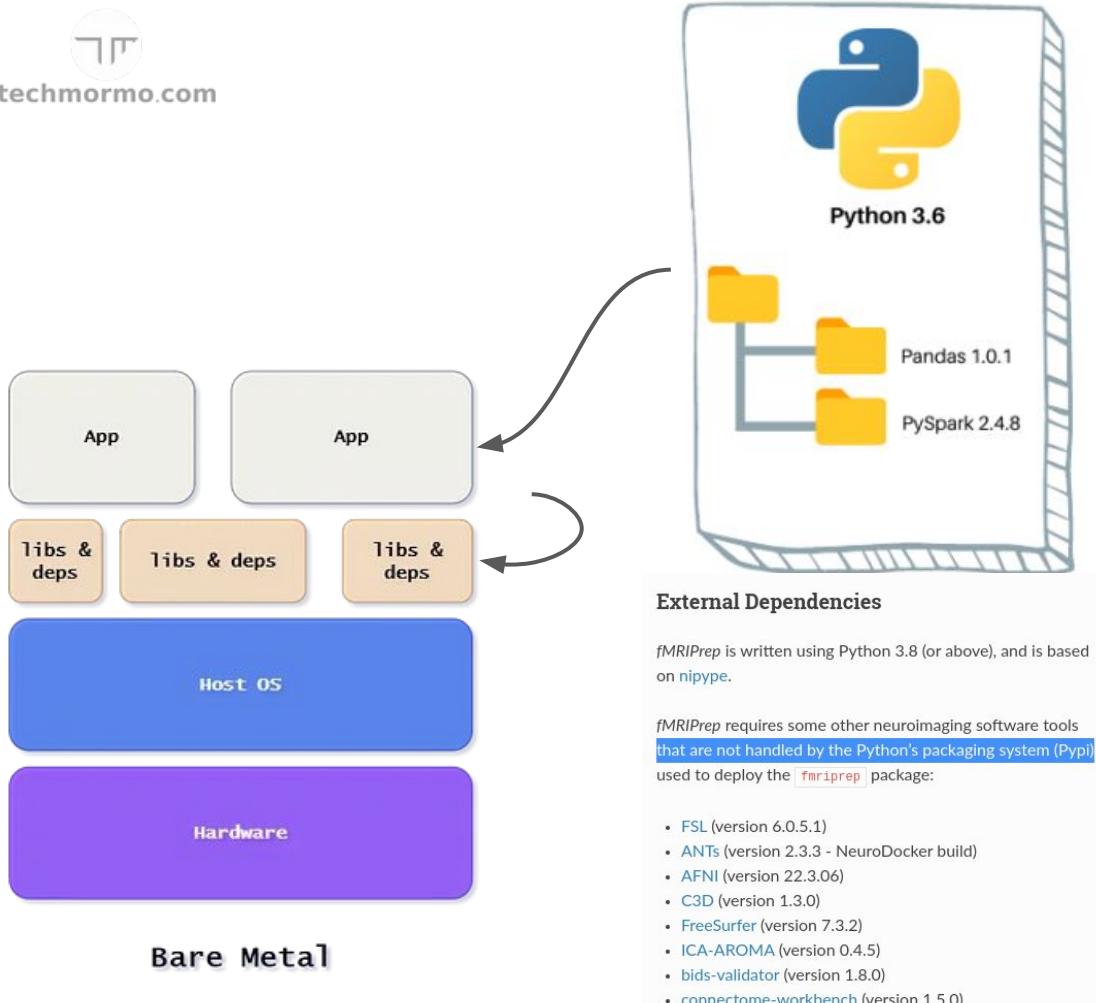


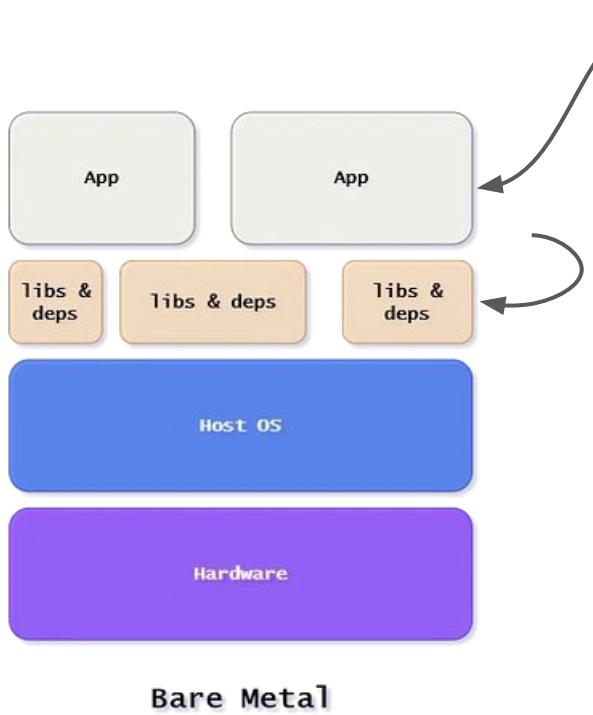
**Bare Metal**



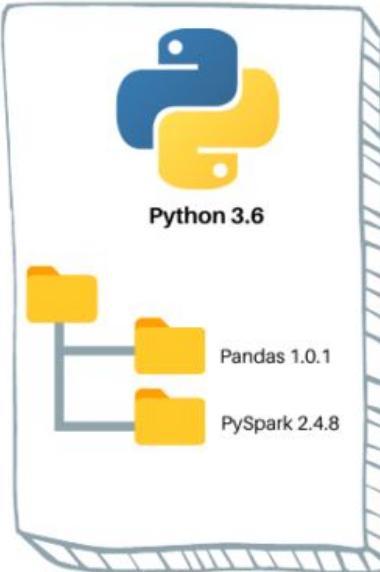
techmormo.com







## Virtual Environment 2



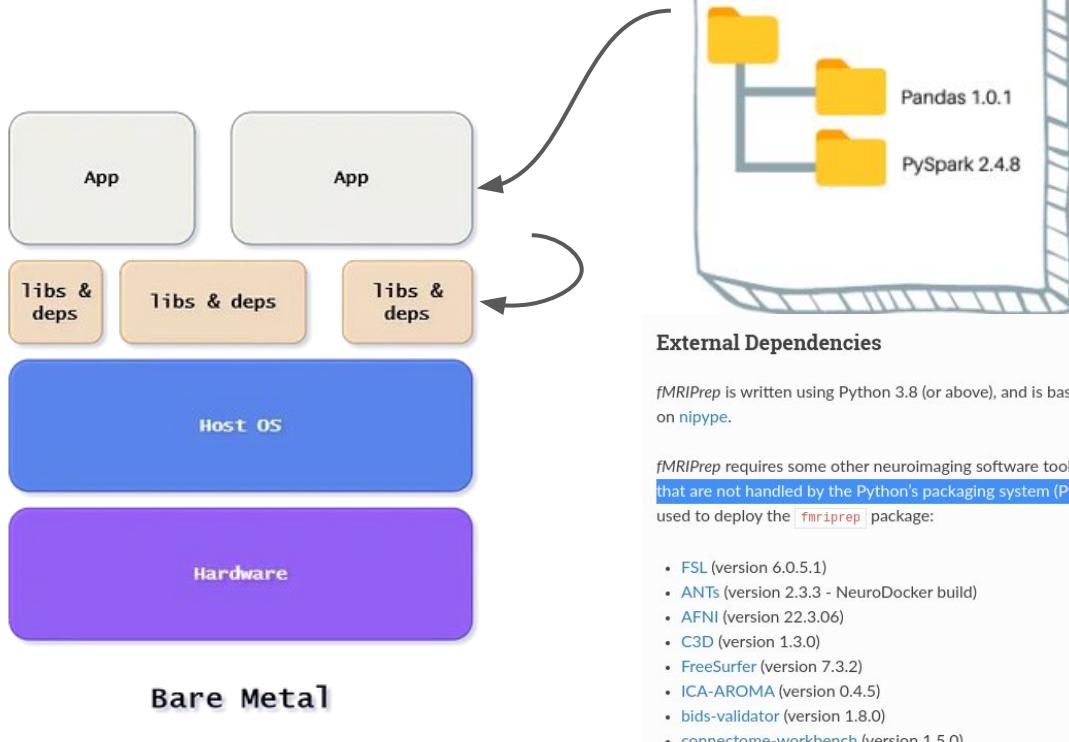
### External Dependencies

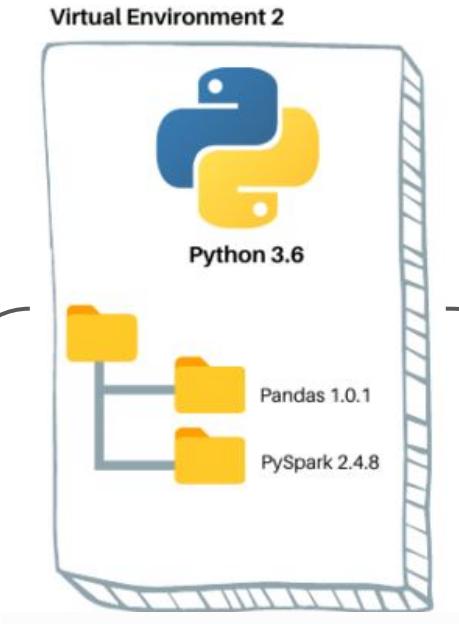
*fMRIPrep* is written using Python 3.8 (or above), and is based on [nipyne](#).

*fMRIPrep* requires some other neuroimaging software tools that are not handled by the Python's packaging system (Pypi) used to deploy the [fmrifprep](#) package:

- FSL (version 6.0.5.1)
- ANTs (version 2.3.3 - NeuroDocker build)
- AFNI (version 22.3.06)
- C3D (version 1.3.0)
- FreeSurfer (version 7.3.2)
- ICA-AROMA (version 0.4.5)
- bids-validator (version 1.8.0)
- connectome-workbench (version 1.5.0)

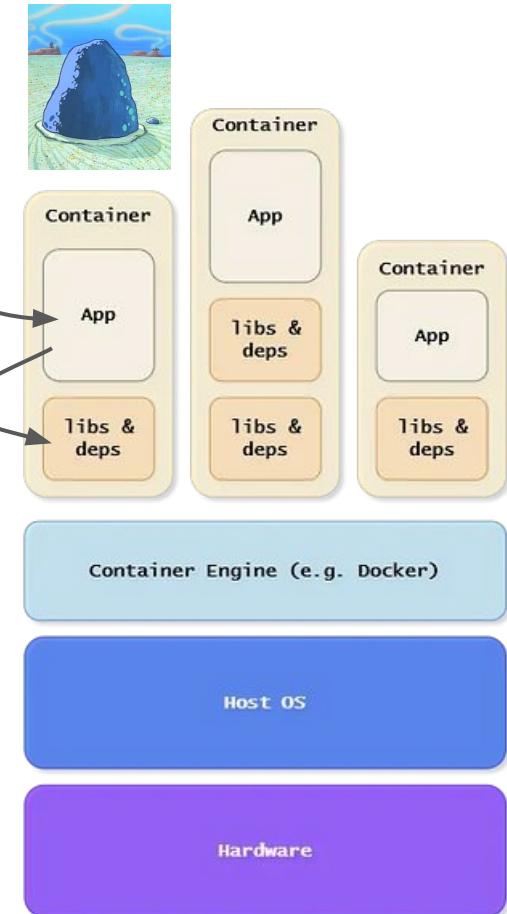






*fMRIPrep* requires some other neuroimaging software tools that are not handled by the Python's packaging system (Pypi) used to deploy the [fMRIprep](#) package:

- FSL (version 6.0.5.1)
- ANTs (version 2.3.3 - NeuroDocker build)
- AFNI (version 22.3.06)
- C3D (version 1.3.0)
- FreeSurfer (version 7.3.2)
- ICA-AROMA (version 0.4.5)
- bids-validator (version 1.8.0)
- connectome-workbench (version 1.5.0)



# Containers

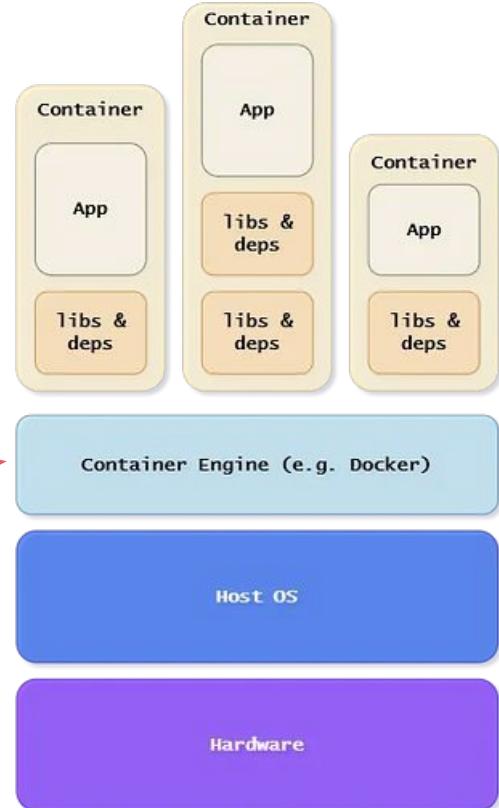
# Containers

## What are they

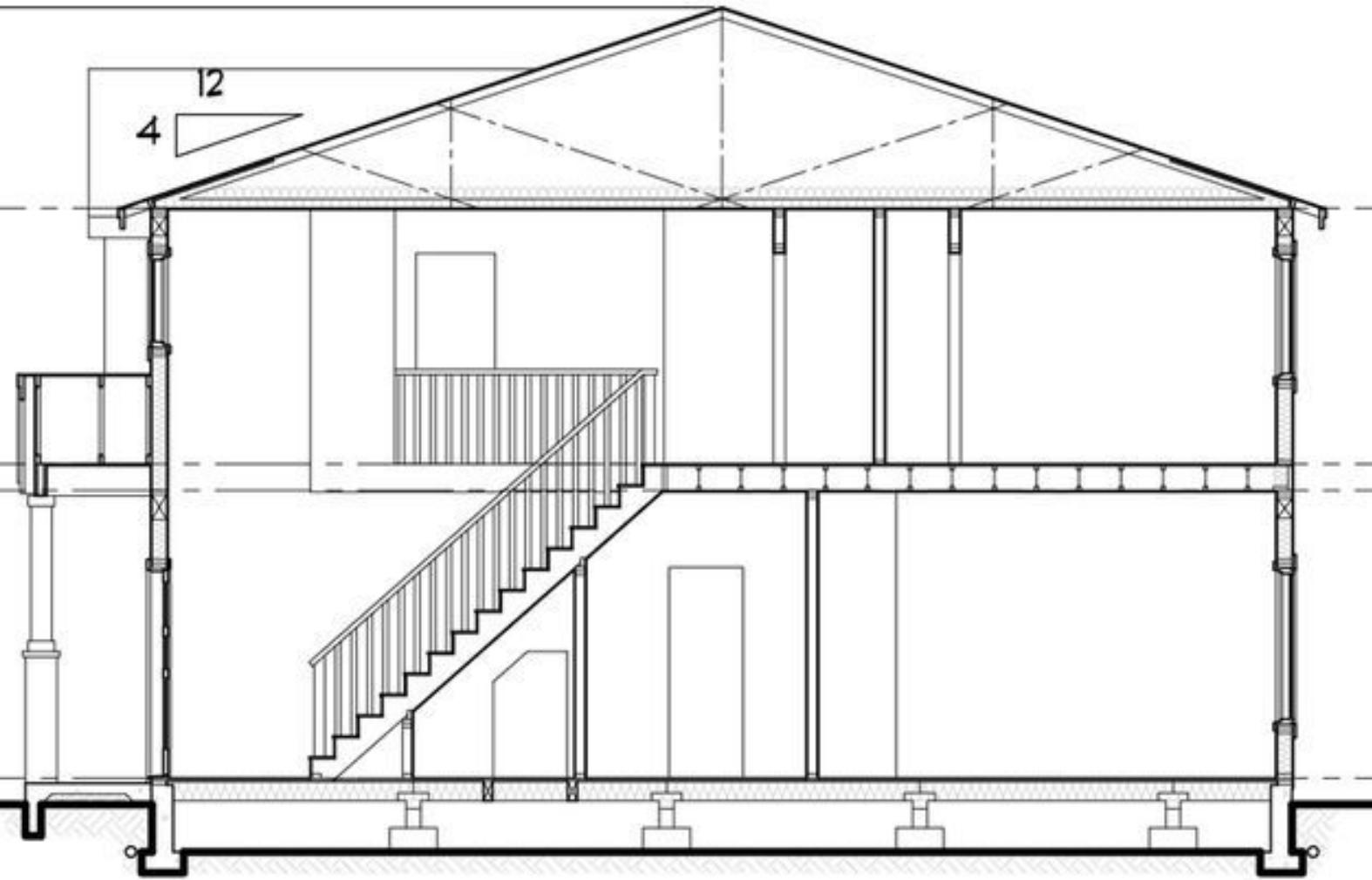
- isolated environments sharing the same kernel / OS -> **(OS virtualization)**
- from the inside, a container looks like a separate computer, can't see outside
- within each container, you can have your desired binary and library dependencies

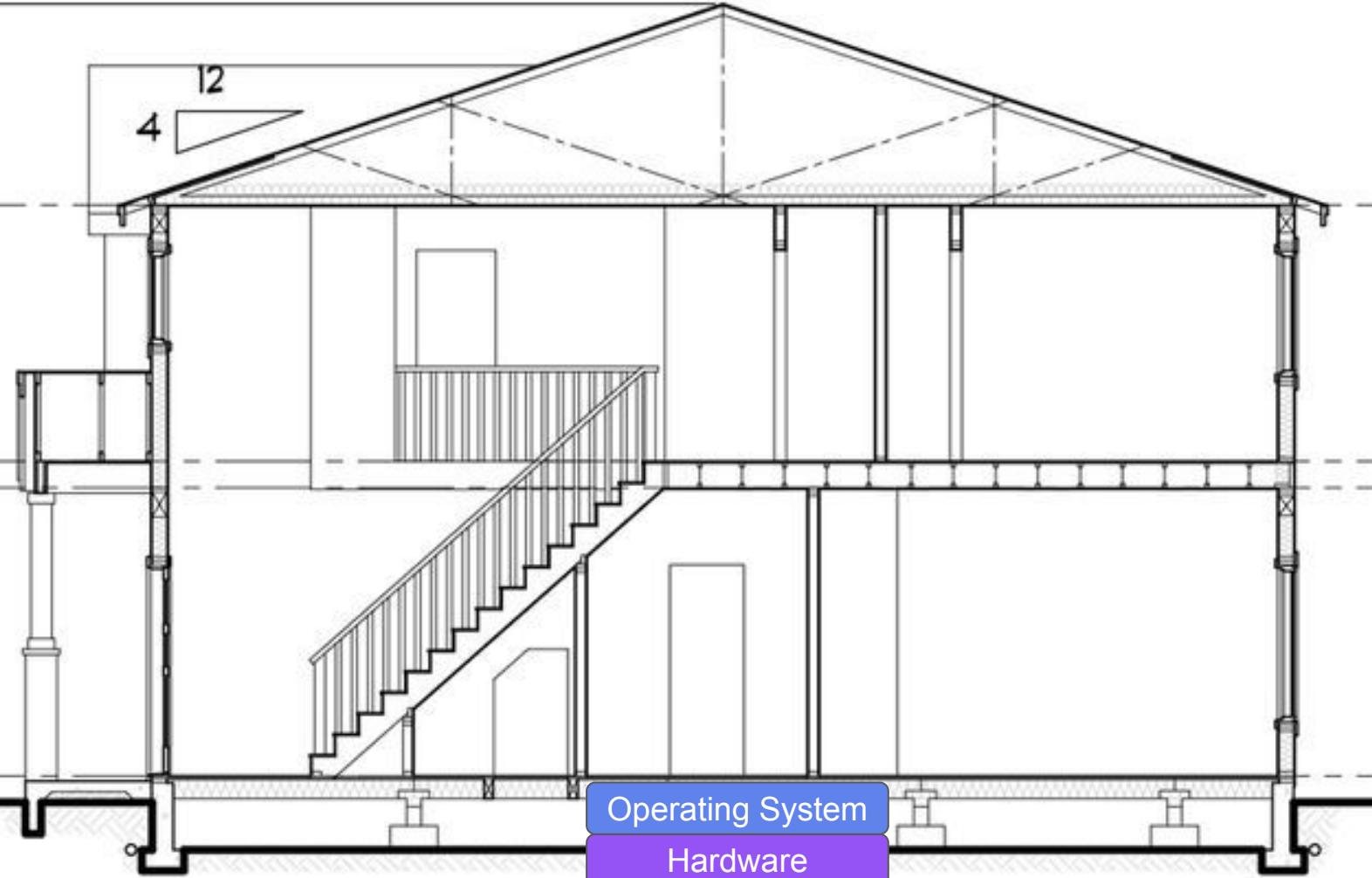
## How do I make one

- use a container implementation
- **docker** is the most widely used
- Singularity is used on supercomputers



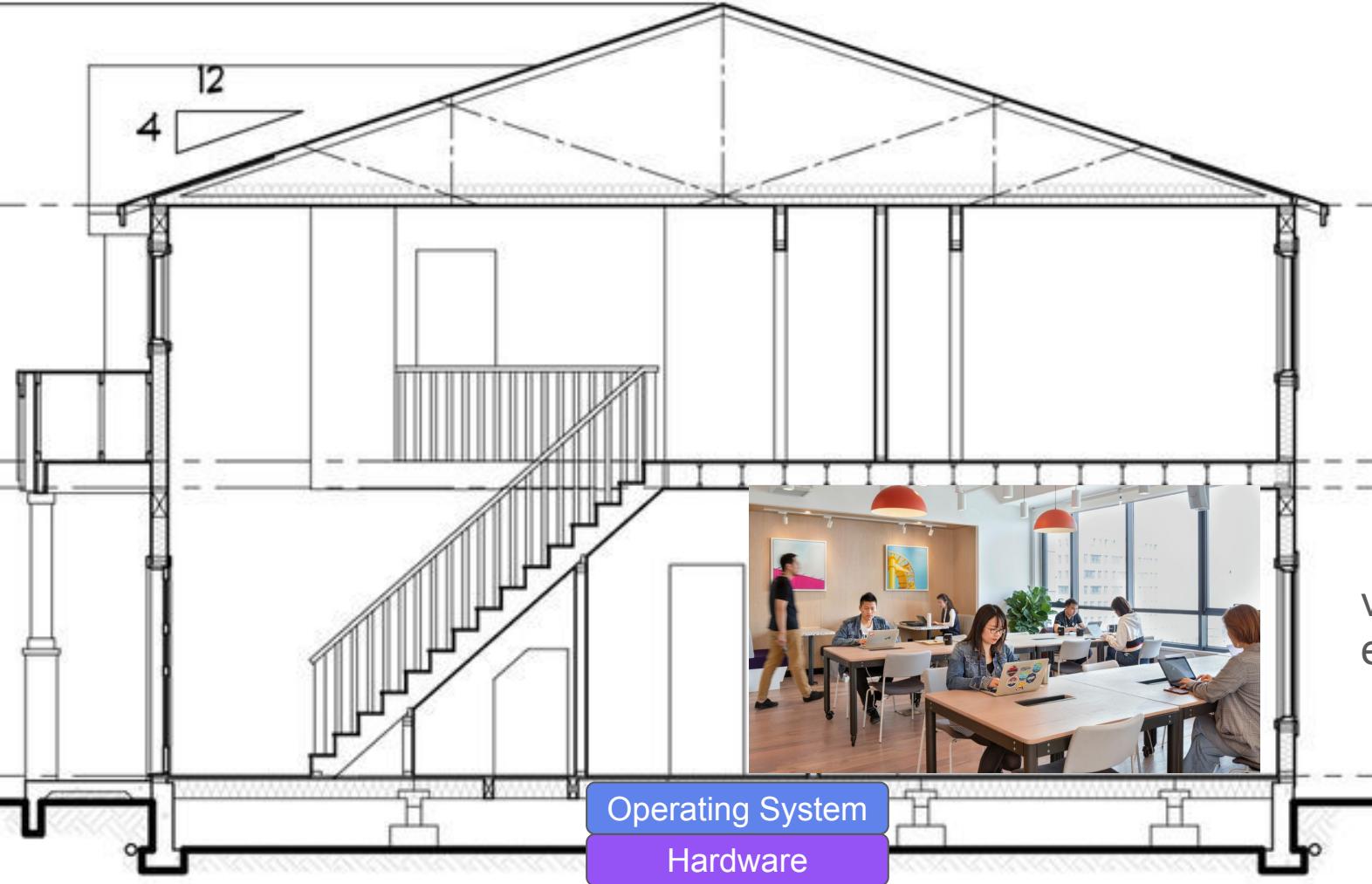
Containers





Operating System

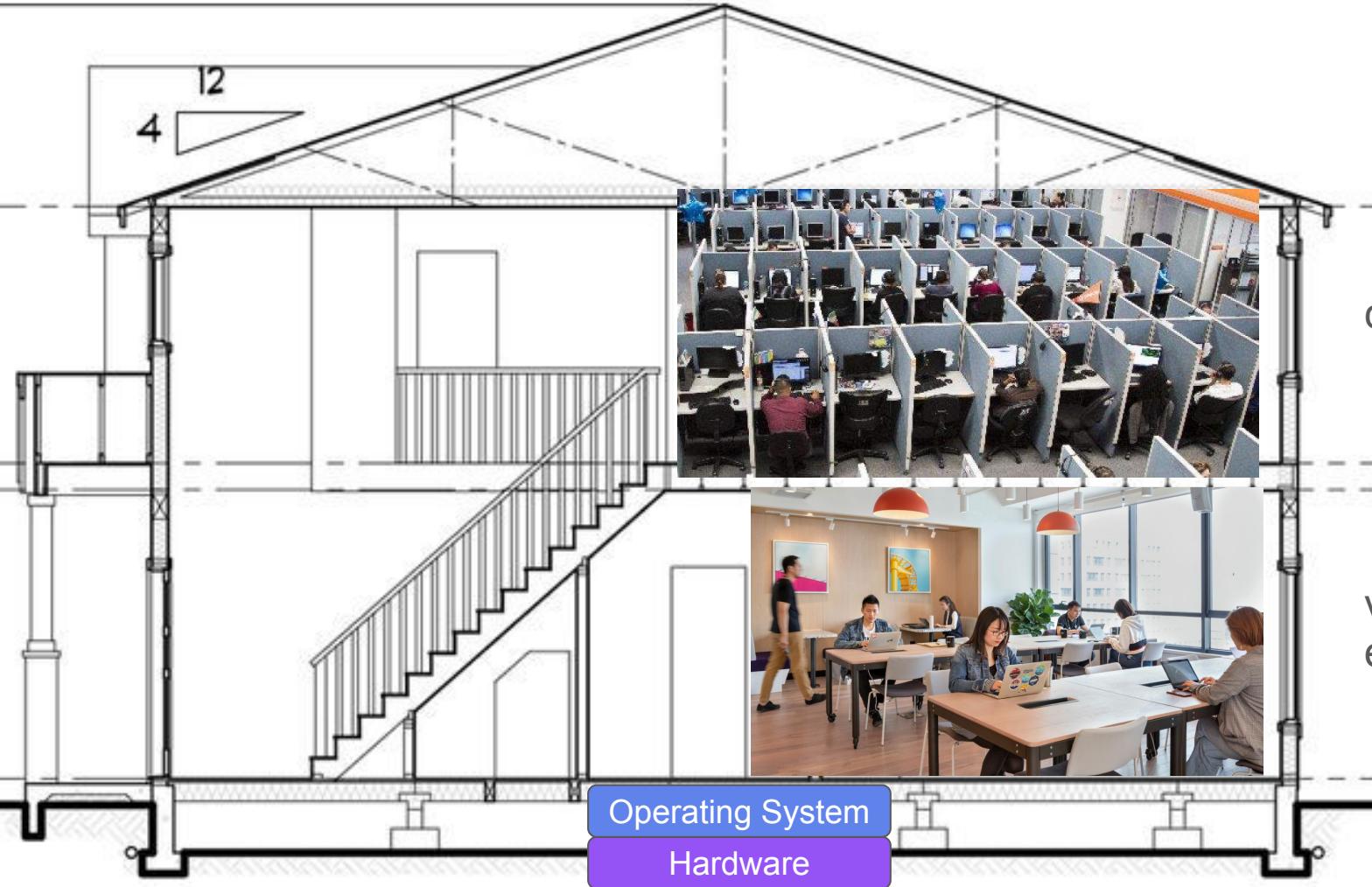
Hardware



virtual  
environments

Operating System

Hardware

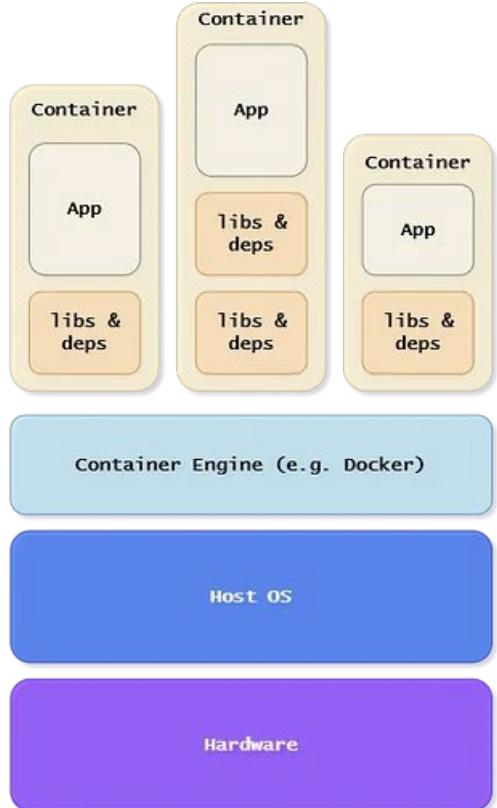


containers

virtual  
environments



Bare Metal



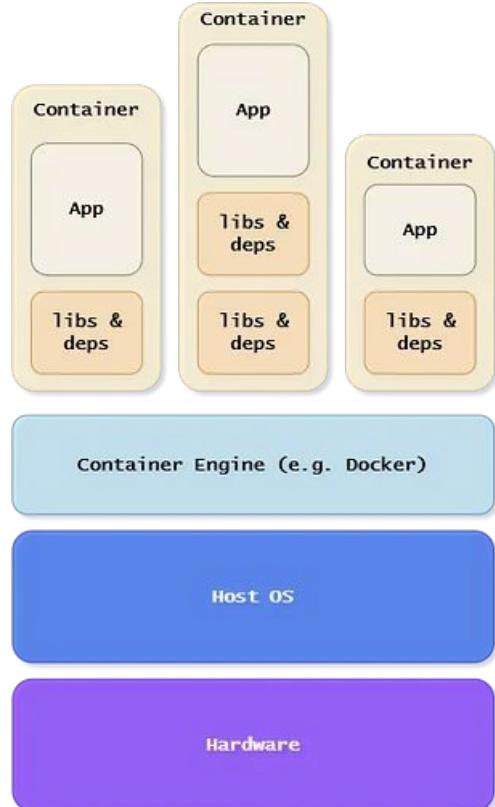
Containers



**Bare Metal**

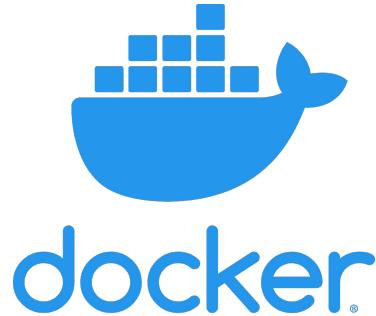


**Virtual Machines**



**Containers**

# What is Docker



## Docker Engine

- a command line program
- gets and builds Docker images and runs Docker containers



## Docker Hub

- a website / web service
- a central repository to store and share Docker container images (commercial)
- other container image registries exist

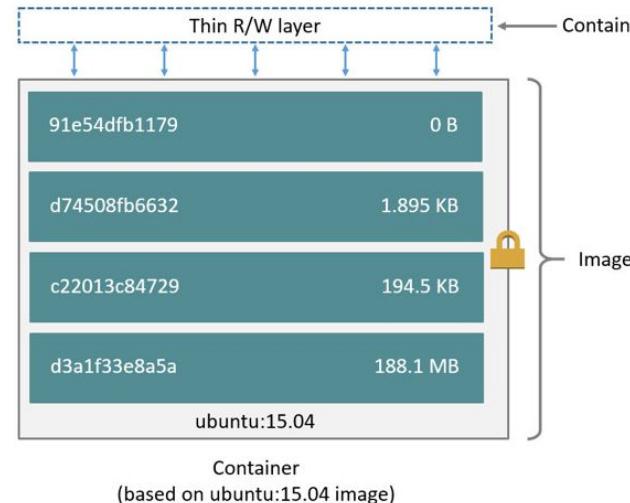
# Docker **image** and **container**: what's the difference

## Docker image

- a **read-only** snapshot of an environment
- changing an image adds more layers
- can be stored on Dockerhub or as a file
- images can share identical layers
- can make your own with a **Dockerfile**

## Docker container

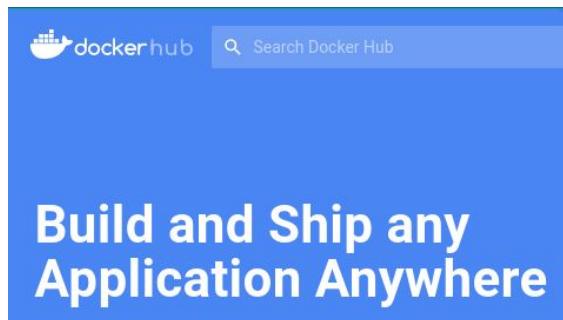
- a **live instance** of a Docker image
- has a thin writable layer that dies with it
- **one image can spawn many containers**



# How can I get my own Docker container going

## Use an existing image

- Dockerhub: a repository of docker images  
<https://hub.docker.com/>
- You can pull an image with `docker pull`



## Build your own image

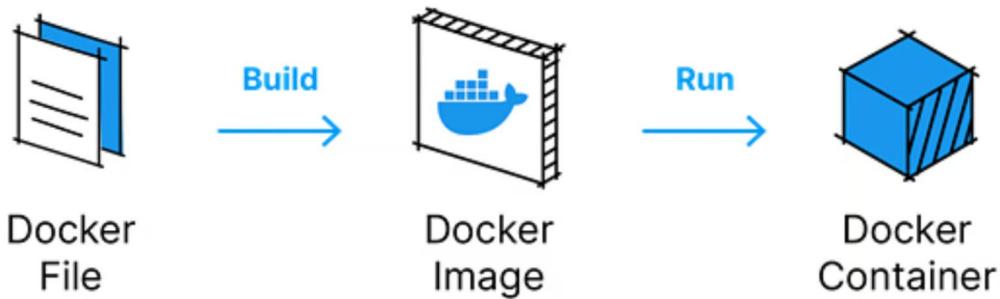
- a `Dockerfile` lets you describe the exact image you want to create
- Start from one image and add to it

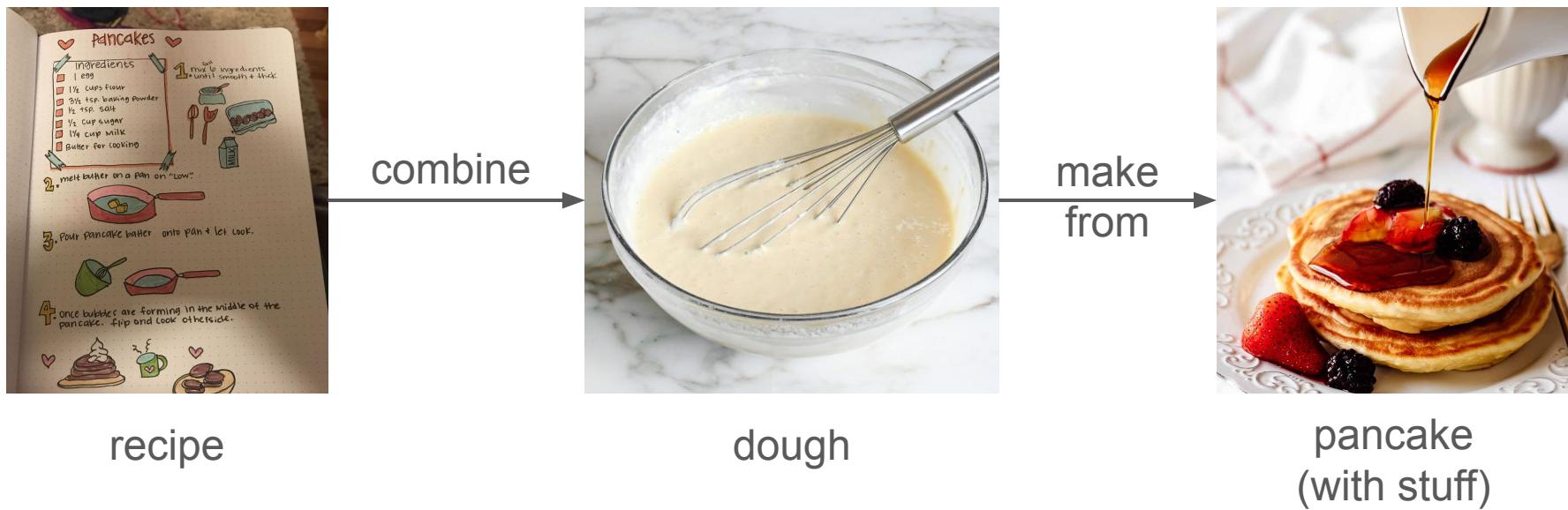
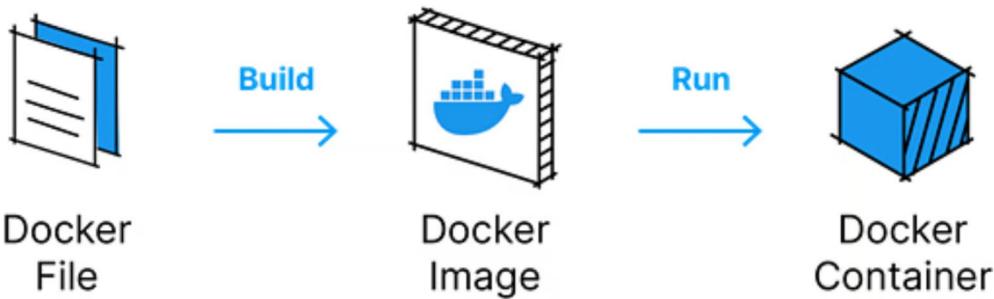
```
# syntax=docker/dockerfile:1
FROM python:3.10-slim-buster
WORKDIR /app
COPY . /app/src
RUN pip install -r /app/src/requirements.txt
RUN pip install --no-cache-dir --no-deps /app/src[all]

ENTRYPOINT [ "bagel" ]
CMD [ "--help" ]
```

A screenshot of a terminal window displaying a Dockerfile. The first line, `FROM python:3.10-slim-buster`, is highlighted with a red rectangular box and a red arrow points from the left towards this box. The Dockerfile contains several other commands: WORKDIR, COPY, RUN, ENTRYPOINT, and CMD.

Let's look at both





# Exercise 1: Run a container from Dockerhub

- Find an image we like: [https://hub.docker.com/\\_/hello-world](https://hub.docker.com/_/hello-world)
- Take a look at it
- Pull the image
- Run the image

# Exercise 1: Run a container from Dockerhub

- Find an image we like: [https://hub.docker.com/\\_/hello-world](https://hub.docker.com/_/hello-world)
- Take a look at it
- Pull the image
- Run the image

Copy and paste to pull this image

`docker pull hello-world`



[View Available Tags](#)

```
surchs@deepthought:~/Documents/docker_place
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
```

# Exercise 1: Run a container from Dockerhub

## Summary

- Dockerhub has images
- Image tags are important
- We can
  - retrieve images with `docker pull`
  - create and immediately run a container from an image with `docker run`

Let's find a more useful image

## Exercise 2: work with a container that has conda

I don't have conda installed on my system. But there is a Docker image. Let's try!

- Find a Docker image: <https://hub.docker.com/r/continuumio/miniconda3/>
- Pick a tag, then pull the image `$ docker pull continuumio/miniconda3:22.11.1-alpine`
- Run it

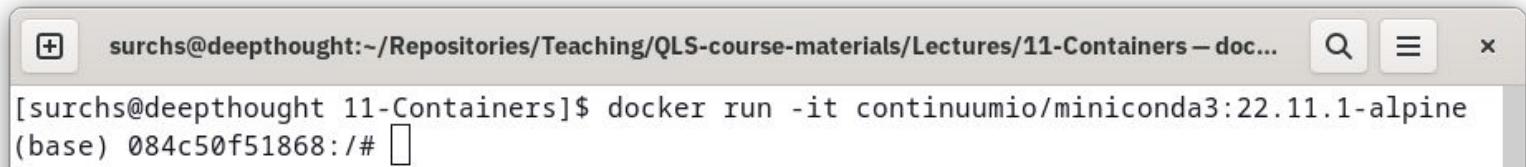
## Exercise 2: work with a container that has conda

I don't have conda installed on my system. But there is a Docker image. Let's try!

- Find a Docker image: <https://hub.docker.com/r/continuumio/miniconda3/>
- Pick a tag, then pull the image \$ docker pull continuumio/miniconda3:22.11.1-alpine
- Run it

Oh, nothing happened?

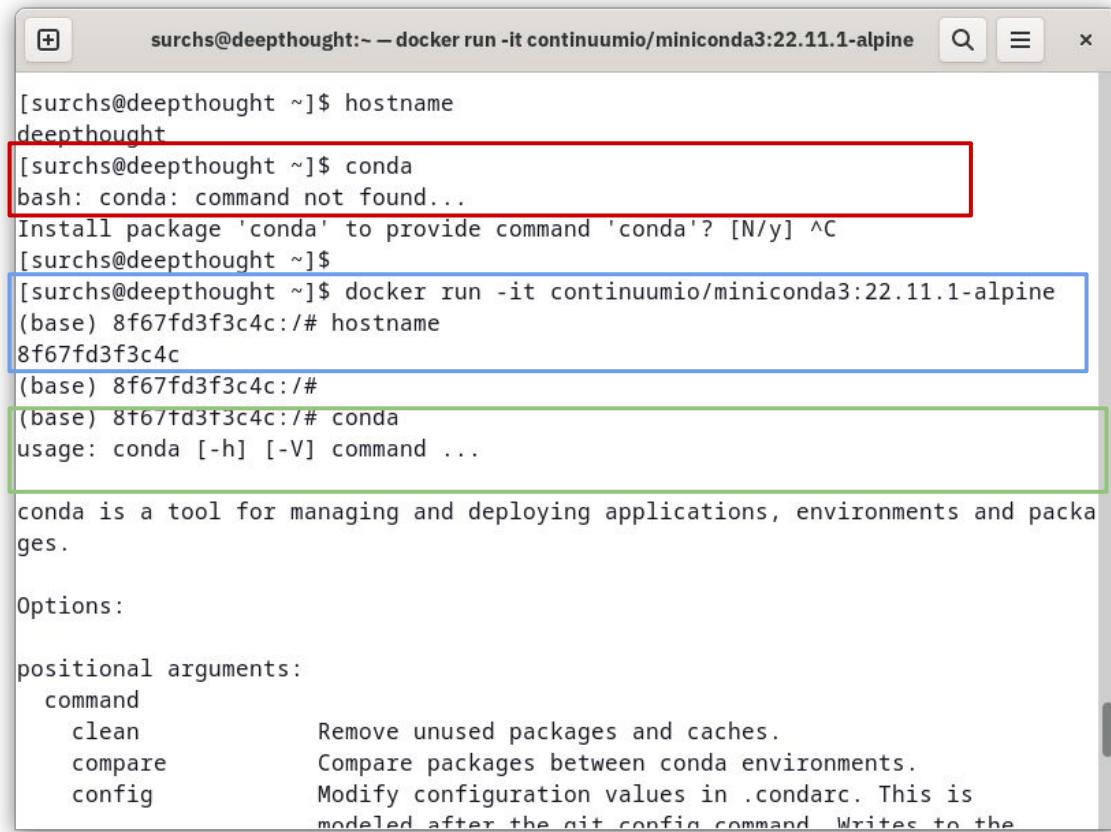
- Let's run it interactively to take a look inside



A screenshot of a terminal window. The title bar says "surchs@deepthought:~/Repositories/Teaching/QLS-course-materials/Lectures/11-Containers – doc...". The command entered is "[surchs@deepthought 11-Containers]\$ docker run -it continuumio/miniconda3:22.11.1-alpine". The terminal prompt "(base) 084c50f51868:/#" is visible at the bottom.

# Looking around inside a container

- no conda on my machine
- starting container changes the look of my terminal and the name of my computer
- inside of the container I have access to conda



The screenshot shows a terminal window with the following session:

```
[surchs@deepthought:~] docker run -it continuumio/miniconda3:22.11.1-alpine
[surchs@deepthought ~]$ hostname
deepthought
[surchs@deepthought ~]$ conda
bash: conda: command not found...
Install package 'conda' to provide command 'conda'? [N/y] ^C
[surchs@deepthought ~]$ docker run -it continuumio/miniconda3:22.11.1-alpine
(base) 8f67fd3f3c4c:/# hostname
8f67fd3f3c4c
(base) 8f67fd3f3c4c:/#
(base) 8f67fd3f3c4c:/# conda
usage: conda [-h] [-V] command ...
conda is a tool for managing and deploying applications, environments and packages.

Options:

positional arguments:
  command
    clean           Remove unused packages and caches.
    compare         Compare packages between conda environments.
    config          Modify configuration values in .condarc. This is
                    modeled after the git config command. Writes to the
```

The terminal window has several colored highlights:

- A red box surrounds the command `conda` and its error message.
- A blue box surrounds the command `hostname` and its output.
- A green box surrounds the command `conda` and its usage information.

# By default the container doesn't see files on the host ...

The screenshot shows a terminal window with a red box highlighting the host environment and a green box highlighting the container environment.

**outside (on host)**

```
[surchs@deepthought docker_place]$ pwd  
/home/surchs/Documents/docker_place  
[surchs@deepthought docker_place]$ ls  
a_file_on_the_host.txt
```

**inside (in container)**

```
[surchs@deepthought docker_place]$ docker run -it continuumio/miniconda3:22.11.1-alpine  
(base) c179244ae50f:/# pwd  
/  
(base) c179244ae50f:/# ls  
bin etc lib media opt root sbin sys usr  
dev home lib64 mnt proc run srv tmp var  
(base) c179244ae50f:/# cd /home/surchs/Documents/docker_place  
bash: cd: /home/surchs/Documents/docker_place: No such file or directory  
(base) c179244ae50f:/#
```

... and the host can't see files on the container ...

The screenshot shows a terminal window with the following session:

```
[surchs@deepthought docker_place]$ docker run -it continuumio/miniconda3:22.11.1-alpine
(base) fa973dee589c:/# ls
bin etc lib media opt root sbin sys usr
dev home lib64 mnt proc run srv tmp var
(base) fa973dee589c:/# touch container_file.txt
(base) fa973dee589c:/# ls
bin lib proc sys
container_file.txt lib64 root tmp
dev media run usr
etc mnt sbin var
home opt srv

(base) fa973dee589c:/#
exit
[surchs@deepthought docker_place]$ ls /container_file.txt
ls: cannot access '/container_file.txt': No such file or directory
[surchs@deepthought docker_place]$ 
```

The terminal window has a green border around the first part of the session (inside the container), and a red border around the last command (outside the container). Labels on the right side identify the regions:

- inside (in container)
- outside (on host)

# ... and files written to a container are tied to it!

```
surchs@deepthought docker_place]$ docker run -it continuumio/miniconda3:22.11.1-alpine
(base) 0fb93cb1903e:/# ls
bin etc lib media opt root sbin sys usr
dev home lib64 mnt proc run srv tmp var
(base) 0fb93cb1903e:/# touch file1.txt
(base) 0fb93cb1903e:/# ls
bin file1.txt lib64 opt run sys var
dev home media proc sbin tmp
etc lib mnt root srv usr
(base) 0fb93cb1903e:/#
exit
[surchs@deepthought docker_place]$ docker run -it continuumio/miniconda3:22.11.1-alpine
(base) 2b690713ac84:/# ls
bin etc lib media opt root sbin sys usr
dev home lib64 mnt proc run srv tmp var
(base) 2b690713ac84:/# ls file1.txt
ls: file1.txt: No such file or directory
(base) 2b690713ac84:/#
exit
[surchs@deepthought docker_place]$
```

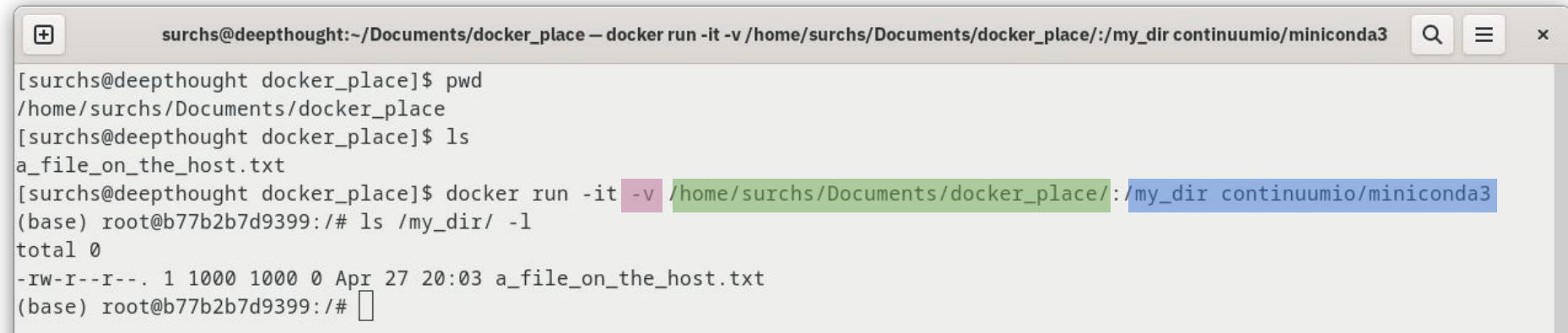
first container instance

second container instance

-> Don't write anything (worth keeping) to a container

# How do we share files between host and container?

- Mount a path on the host to a path in the container



A screenshot of a terminal window titled "surchs@deepthought:~/Documents/docker\_place". The terminal shows the following command being run:

```
surchs@deepthought:~/Documents/docker_place$ docker run -it -v /home/surchs/Documents/docker_place:/my_dir continuumio/miniconda3
```

The terminal then lists the contents of the directory:

```
[surchs@deepthought docker_place]$ pwd  
/home/surchs/Documents/docker_place  
[surchs@deepthought docker_place]$ ls  
a_file_on_the_host.txt  
[surchs@deepthought docker_place]$ docker run -it -v /home/surchs/Documents/docker_place:/my_dir continuumio/miniconda3  
(base) root@b77b2b7d9399:/# ls /my_dir/ -l  
total 0  
-rw-r--r--. 1 1000 1000 0 Apr 27 20:03 a_file_on_the_host.txt  
(base) root@b77b2b7d9399:/#
```

- The bind-mount will be created in the container, even if it exists already!
- Make sure you provide a path (starts with / or ./) - or use --mount

# Exercise 2: work with a container that has conda

## Summary

- We can connect to an interactive shell in a container with `docker run -it`
- By default the container cannot see or write the filesystem of the host
- We can “mount” a path on the host into the container with

```
docker run -v /host/path:/container/path OR
```

```
docker run --mount type=bind,source=/host/path,target=/container/path
```

- It's a bad idea to write into the container directly

So how do I make persistent changes to the image?

# Exercise 3: Run a Dockerized tool on local data

We've explored how a Docker container behaves on the inside. Let's follow the more "typical" steps to run a tool available as a Docker image.



inside



outside

## Exercise 3: Run a Dockerized tool on local data

Let's try running the [BIDS validator](#) on a test dataset ds006, to see if the dataset complies with the Brain Imaging Dataset Structure.

1. Pull the [bids/validator](#) Docker image tagged latest, and list the images on your system to confirm that bids/validator is there

```
docker pull bids/validator:latest  
docker images
```

# Exercise 3: Run a Dockerized tool on local data

Recall: `docker run [options-for-docker] <imagename> <command>`

Let's print the help text for the bids/validator:

```
docker run --rm bids/validator --help
```

Output:

```
Usage: bids-validator <dataset_directory> [options]
```

...

# Exercise 3: Run a Dockerized tool on local data

Recall: `docker run [options-for-docker] <imagename> <command>`

Let's print the help text for the bids/validator:

```
docker run --rm bids/validator --help
```

Output:

```
Usage: bids-validator <dataset_directory> [options]
```

...

...How do I run other validator commands?

```
docker run --rm bids/validator <dataset_directory> [options]
```

# Exercise 3: Run a Dockerized tool on local data

```
docker run --rm bids/validator <dataset_directory> [options]
```

Our test dataset is located **on our host machine** under:  
QLS-course-materials/Lectures/2024/data/ds006

2. Run the bids/validator image again, but provide this dataset directory to validate.

HINTS:

- **Mount** the dataset directory into the container for it to find the dataset (remember: the path must start with / or ./ !)

# Exercise 3: Run a Dockerized tool on local data

```
docker run --rm bids/validator <dataset_directory> [options]
```

Our test dataset is located **on our host machine** under:  
QLS-course-materials/Lectures/2024/data/ds006

2. Run the bids/validator image again, but provide this dataset directory to validate.

HINTS:

- Mount the dataset directory into the container for it to find the dataset (remember: the path must start with / or ./ !)
- -v /path/to/data/on/host:/path/to/mount/data/in/container

# Exercise 3: Run a Dockerized tool on local data

```
docker run --rm bids/validator <dataset_directory> [options]
```

Our test dataset is located **on our host machine** under:  
QLS-course-materials/Lectures/2024/data/ds006

2. Run the bids/validator image again, but provide this dataset directory to validate.

HINTS:

- Mount the dataset directory into the container for it to find the dataset (remember: the path must start with / or ./ !)
- -v /path/to/data/on/host:/path/to/mount/data/in/container  
(from data/ directory)

```
docker run --rm -v ./ds006:/data bids/validator /data
```

# Exercise 3: Run a Dockerized tool on local data

The BIDS validator returns errors [ERR] because the test dataset contains empty data files.

3. To make the validator happy, run the Docker image again on the dataset, but this time add **command arguments** to:

- Ignore warnings
- Ignore NIfTI header content

HINT:

- Print the help text again to see all command options

# Exercise 3: Run a Dockerized tool on local data

The BIDS validator returns errors [ERR] because the test dataset contains empty data files.

3. To make the validator happy, run the Docker image again on the dataset, but this time add **command arguments** to:

- Ignore warnings
- Ignore NIfTI header content

HINT:

- Print the help text again to see all command options

```
docker run --rm -v ./ds006:/data bids/validator /data  
--ignoreNiftiHeaders --ignoreWarnings
```

# Apptainer (singularity)



# Apptainer is the container solution for HPCs

On shared systems (like a supercomputer), you shouldn't / can't use Docker

- Docker isn't as isolated as a VM
- By default you run docker with root privileges and are root inside a container
- A malicious actor can escalate privileges and “break out” of a container

**Apptainer (formerly Singularity) is a container solution in these cases**

Singularity<sup>[1]</sup> is open source software created by Berkeley Lab:

- as a **secure way** to use Linux containers on Linux multi-user clusters,
- as a way to enable users to have **full control of their environment**, and,
- as a way to **package scientific software** and deploy such to *different* clusters having the *same* architecture.

i.e., it provides **operating-system-level virtualization** commonly called *containers*.

# Build images with Docker, run them with Apptainer

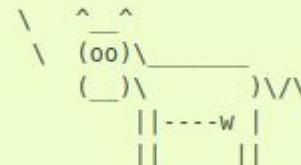
1. Pull an image from Dockerhub and create a local **SingularityImageFile**

```
$ apptainer pull docker://sylabsio/lolcow
```

2. Run the Singularity File using **apptainer run**

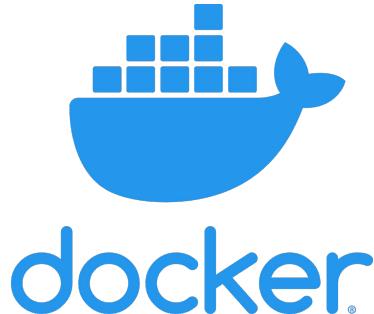
```
$ apptainer run lolcow_latest.sif
```

```
< Mon Aug 16 13:01:55 CDT 2021 >
```



# Container Summary

- Docker **images** are **read-only snapshots**, you can find them on Dockerhub
- Images have tags (or version), that you should specify when pulling
- A Docker **container** is an isolated, **live instance** of an image
- Docker **containers** have their own file system, but this is **not persistent**
- With **volumes** we can **expose directories** on the host to the container
- We can **build** our own images on top of existing ones using a **Dockerfile**
- Use **Apptainer** to run Docker images on a compute cluster



Docker engine

```
1 # our base image
2 FROM alpine:3.5
3
4 # Install python and pip
5 RUN apk add --update py2-pip
6
7 # upgrade pip
8 RUN pip install --upgrade pip
9
10 # install Python modules needed by the Python app
11 COPY requirements.txt /usr/src/app/
```

Dockerfile



Docker image registry

# There are tools to help make Dockerfiles

## Welcome to Neurodocker!

*Neurodocker* is a command-line program that generates custom Dockerfiles and Singularity recipes for neuroimaging and minimizes existing containers. Its purpose is to make it easier for scientists (and others) to easily create reproducible computational environments.

(This requires having [Docker](#) installed)

```
neurodocker generate docker --pkg-manager apt \
    --base-image neurodebian:buster \
    --ants version=2.3.4 \
    --miniconda version=latest conda_install="nipype notebook" \
    --user nonroot
```

# Additional Resources

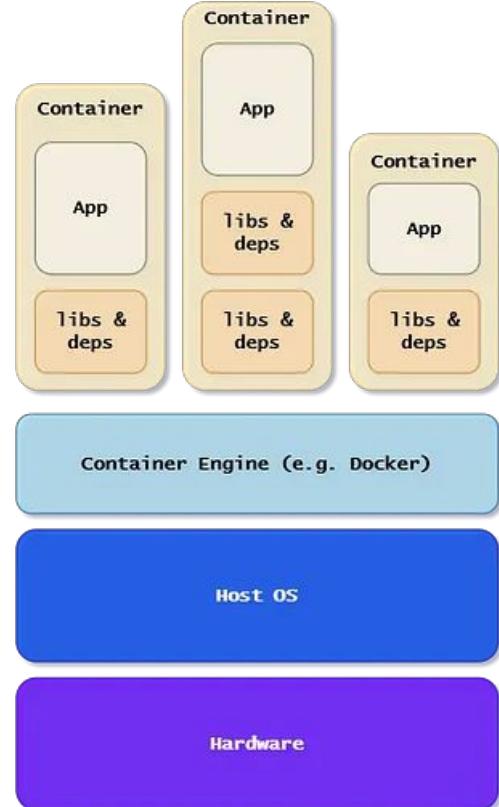
- Docker tutorial <https://github.com/docker/labs>
- Neurohackweek container course  
<https://neurohackweek.github.io/docker-for-scientists/>
- The Turing Way on reproducible research environments  
<https://the-turing-way.netlify.app/reproducible-research/renv.html>



**Bare Metal**



**Virtual Machines**



**Containers**