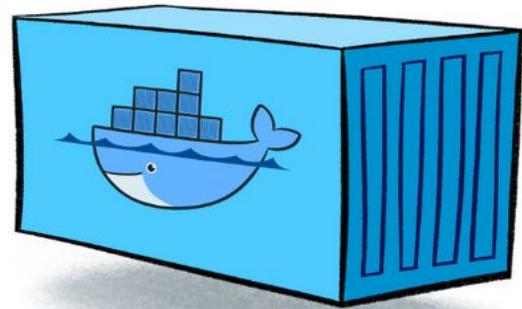


Containerization

QLSC 612 - May 2025
Alyssa Dai

Teaching material adapted from Sebastian Urchs



Containers?



What we will cover

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



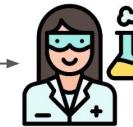
1: to share with colleagues



Document your software environment



1: to share with colleagues



[neurodatascience/cog-align](#) Private

Code Issues Pull requests Actions Security Insights Settings

1 main · 12 branches · 9 tags Go to file Add file · Code

enriqueperez Update fMRIbrain berpict · 7 commits · 16 hours ago

cogalign · 1 commit · 7 months ago

experiments · 1 commit · 7 months ago

figures · 1 commit · 7 months ago

glibcore · 1 commit · 16 months ago

LICENSE · Initial commit · 2 years ago

README.md · Rename Inslightalign -> cog-align · 16 months ago

RPT.py · initial replicate with bugs · 2 years ago

setup.py · Renews Inslightalign -> cogalign · 16 months ago

README.rst

Code Issues Pull requests

cog-align

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 Bazeille*, Elizabeth DuPuis*, Jean-Baptiste Poline, & Bertrand Thirion.
doi: 10.1101/2020.12.07.415000

Requirements

Dependencies :

- `fmrialign`
- `ribabelib<=0.3`
- `humpy<=1.18`
- `anatomicals`
- `perceval`
- `copy`
- `scikit-learn`

Installation

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

`git clone https://github.com/neurodatascience/cog-align.git`

About

Code for the "alignment for cognition" project

- README
- `BSD-3-Clause license`
- 0 stars
- 0 watching
- 0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Push your first package

Contributors 4

enriqueperez Elizabeth DuPuis
thomasbazeille Thomas Bazeille
bertrandthirion Bertrand Thirion
jpoline Jean-Baptiste Poline

Languages

• Python 100.0%

🔗 Requirements

Dependencies

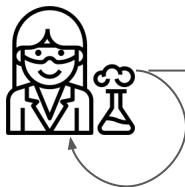
- fmralign
 - nibabel>=3.1
 - numpy>=1.18
 - matplotlib
 - pandas
 - scipy
 - scikit-learn

Installation

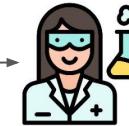
First, make sure you have installed all the dependencies listed above. Then you can install cog-align by running the following commands:

```
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!

Screenshot of a GitHub repository page for "neurodatascience/cog-align".

Code tab: Shows 12 branches and 1 tag. A commit from "eudupe" on Dec 16, 2021, is highlighted: "Update fullbrain_bspf" (7 months ago). Other commits include "fixup functional deleted by mistake" (7 months ago), "experiments" (7 months ago), "Update fullbrain_bspf" (7 months ago), "Add gligner" (16 months ago), "Issue comment" (2 years ago), "Rename fslalignbrain -> cogalign" (16 months ago), "Initial replication with bugs" (16 months ago), and "Rename fslalignbrain -> cogalign" (16 months ago).

About tab: Shows 211 commits, 1 PR, 12 issues, and 1 pull request. It includes sections for "Code", "About", "Releases", and "Packages".

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 .
```

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 .
```

<https://github.com/neurodatascience/cog-align>

Don't use the same environment for all projects



Changing dependencies may do unexpected things

- A. Your updated the dependencies of an existing project

Don't use the same environment for all projects

Changing dependencies may do unexpected things

mriqc / pyproject.toml

fmriprep / pyproject.toml

Code	Blame	235 lines (207 loc)
18]	22 dependencies = [
19 dependencies = [23 "acres >= 0.2.0",	
20 "acres",	24 "looseversion >= 1.3",	
21 "dipy >= 1.10.0",	25 "nibabel >= 4.0.1",	
22 'importlib_resources; p	26 "nipype >= 1.8.5",	
23 "markupsafe ~= 2.0.1",	27 "nireports >= 24.1.0",	
24 "matplotlib",	28 "nitime >= 0.9",	
25 "migas >= 0.4.0",	29 "nitransforms >= 24.1.1",	
26 "mriqc-learn",	30 "niworkflows >= 1.12.2",	
27 "nibabel >= 3.0.1",	31 "numpy >= 1.24",	
28 "nilearn >= 0.5.1",	32 "packaging >= 24",	
29 "nipype ~= 1.4",	33 "pandas >= 1.2",	
30 "nireports ~= 24.0.2",	34 "psutil >= 5.4",	
31 "nitransforms ~= 24.0",	35 "pybids >= 0.16",	
32 "niworkflows ~= 1.10.1",		
33 "numpy >= 1.20"		

- A. Your updated the dependencies of an existing project
- B. Two projects use the same environment but need different versions of some dependencies

Reminder: 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 ~]$ ]
```

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.

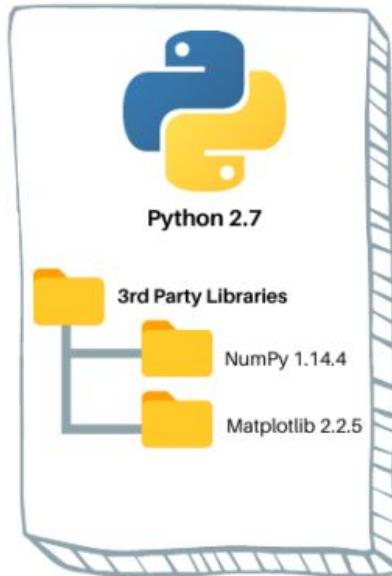


Isolate environments to handle different requirements

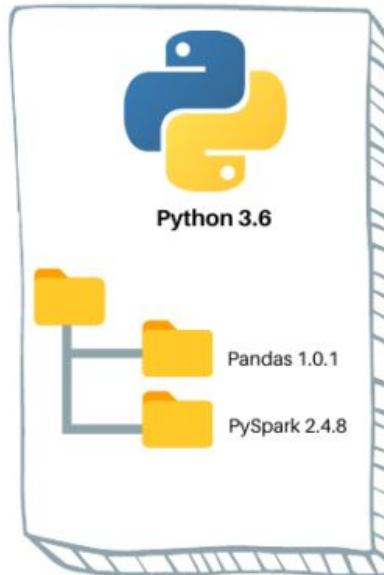


Isolate Python environments

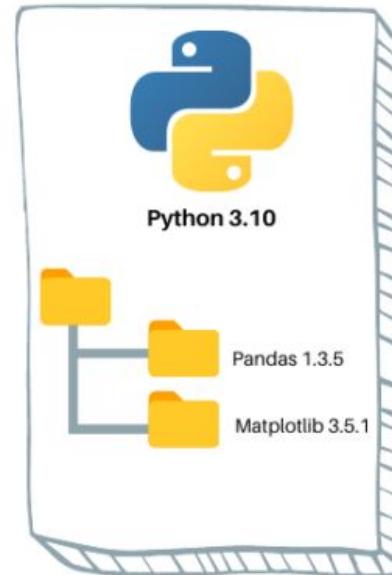
Virtual Environment 1



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 [nipype](#).

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)

- Not all dependencies are supported in 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





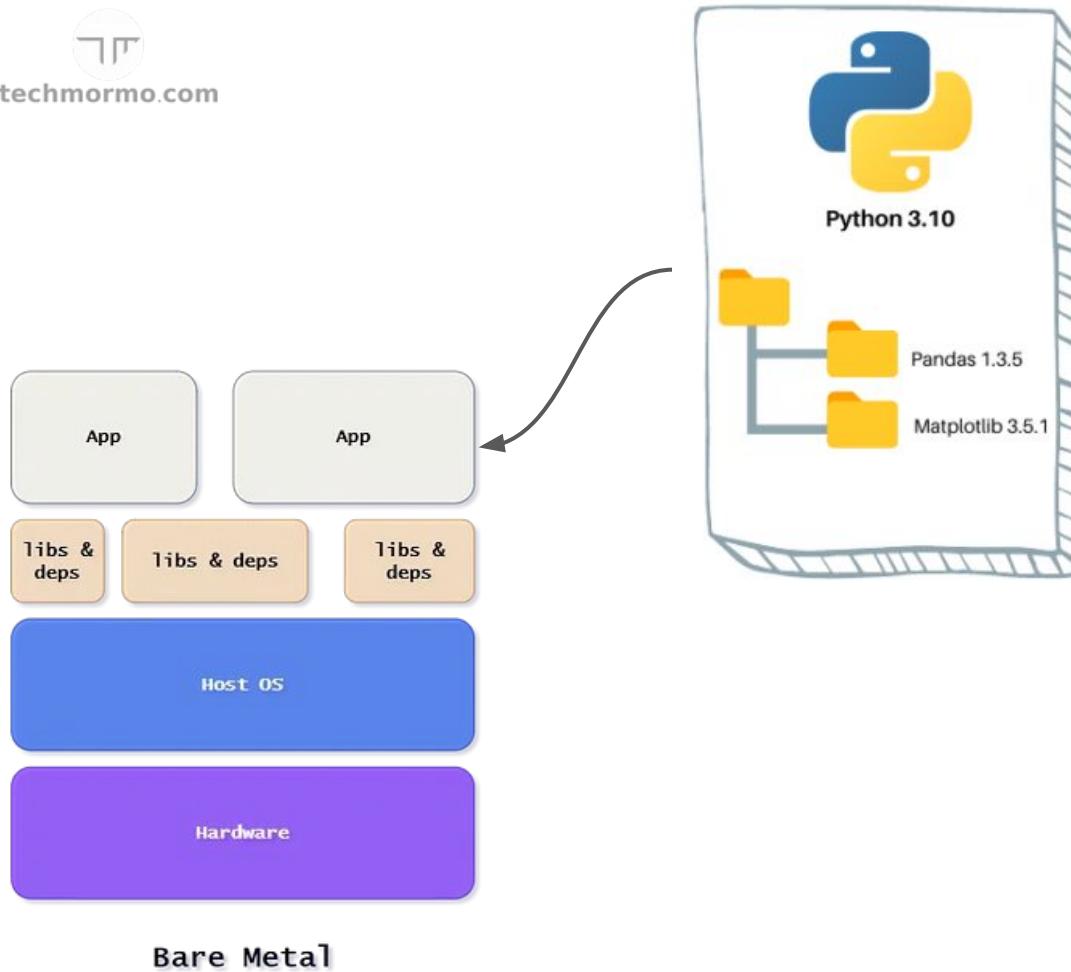
techmormo.com



Bare Metal

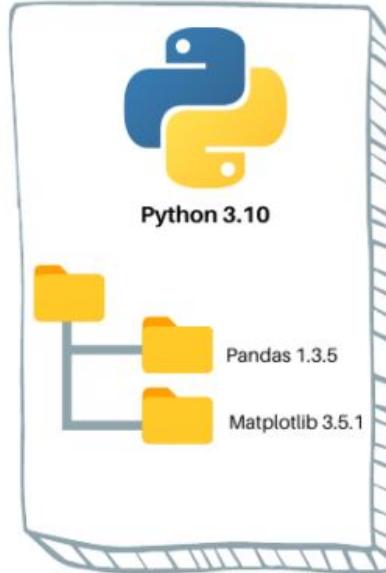


techmormo.com





Virtual Environment 3

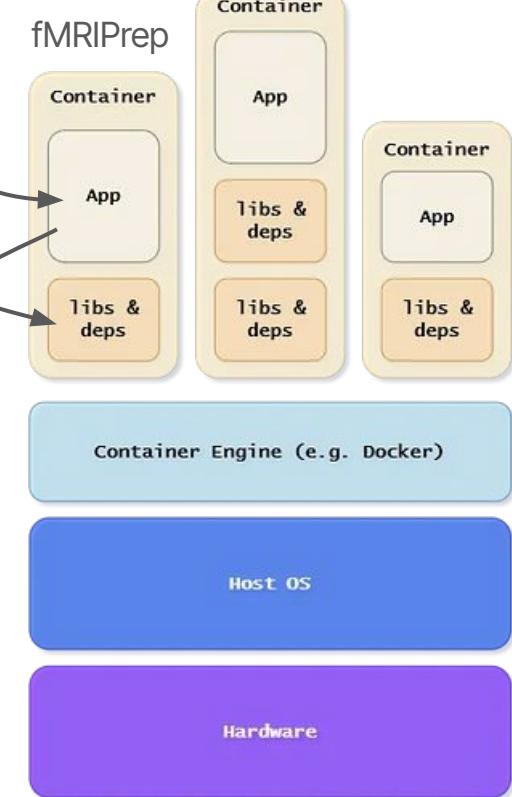
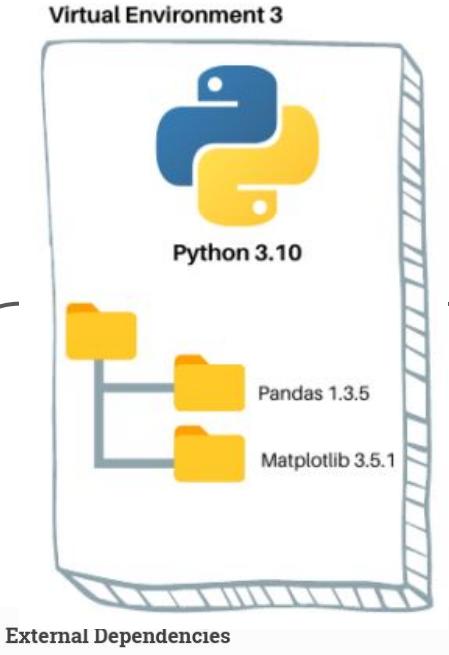
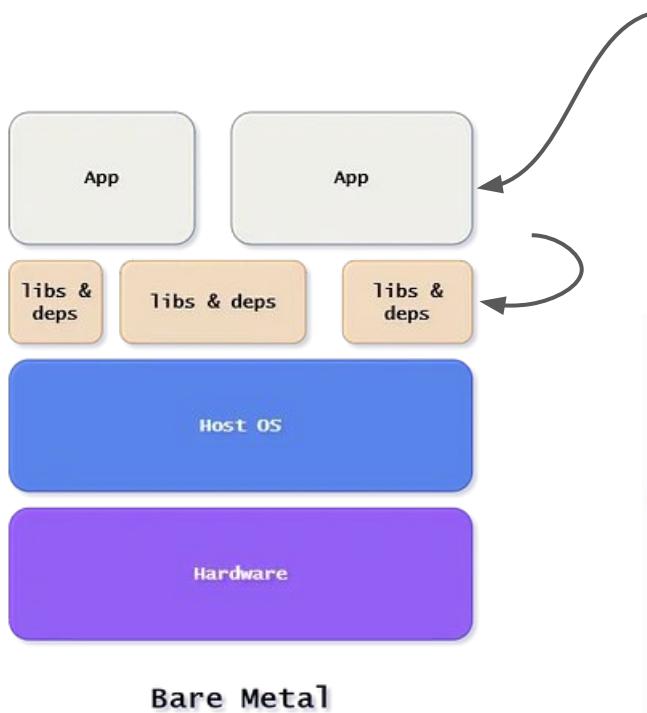


External Dependencies

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

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

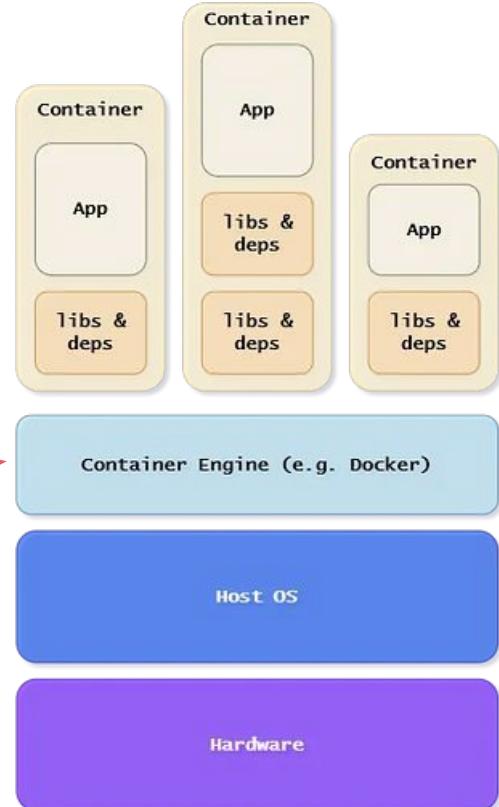
Containers

What are they

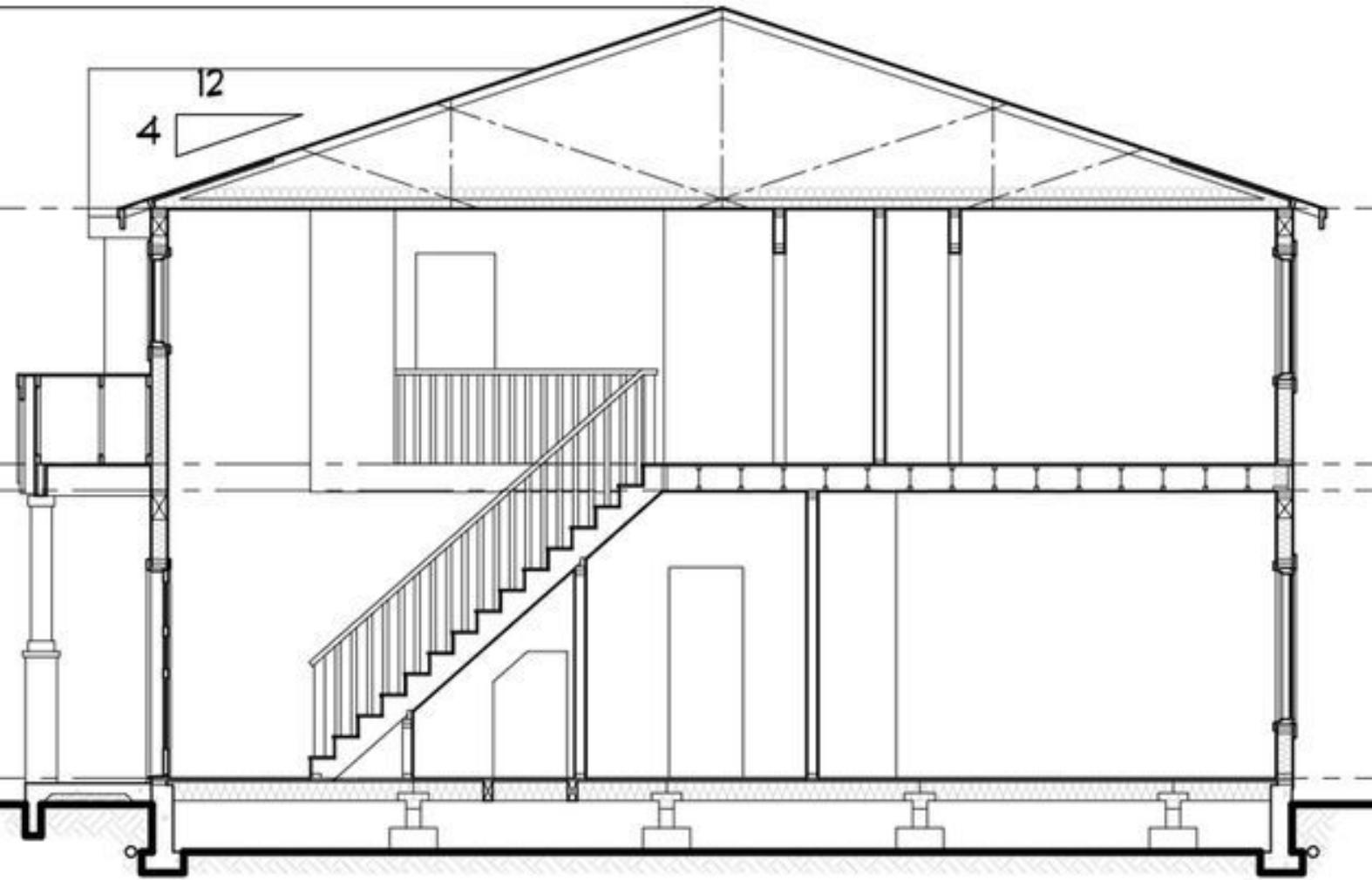
- isolated environments sharing the same host kernel / OS → **(OS virtualization)**
- from the inside, a container looks like a separate computer, can't see outside
- within each container, you can have your desired binary and library dependencies

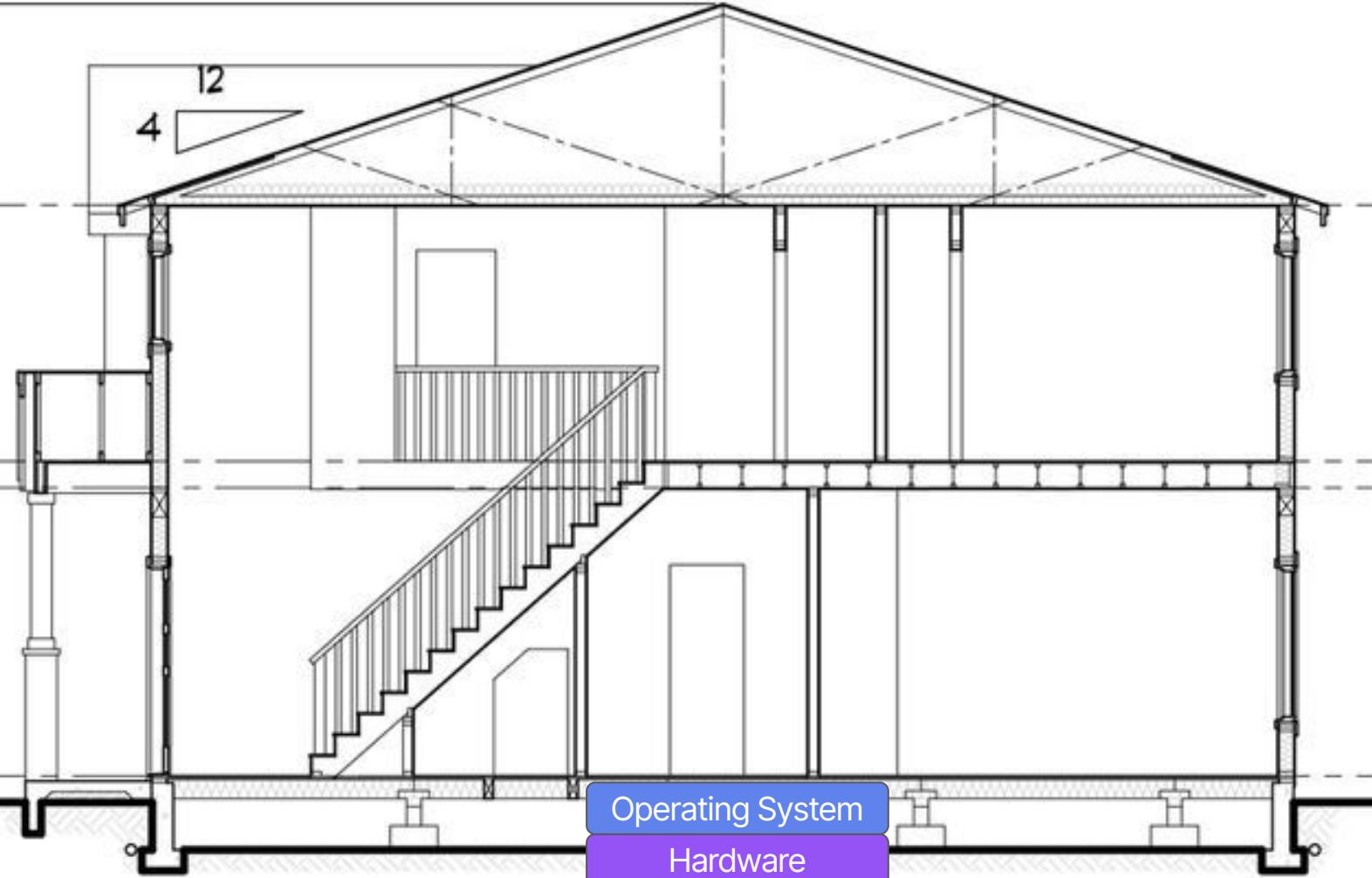
How do I make one

- use a container engine
- **Docker** is the most widely used
- Singularity is used on supercomputers



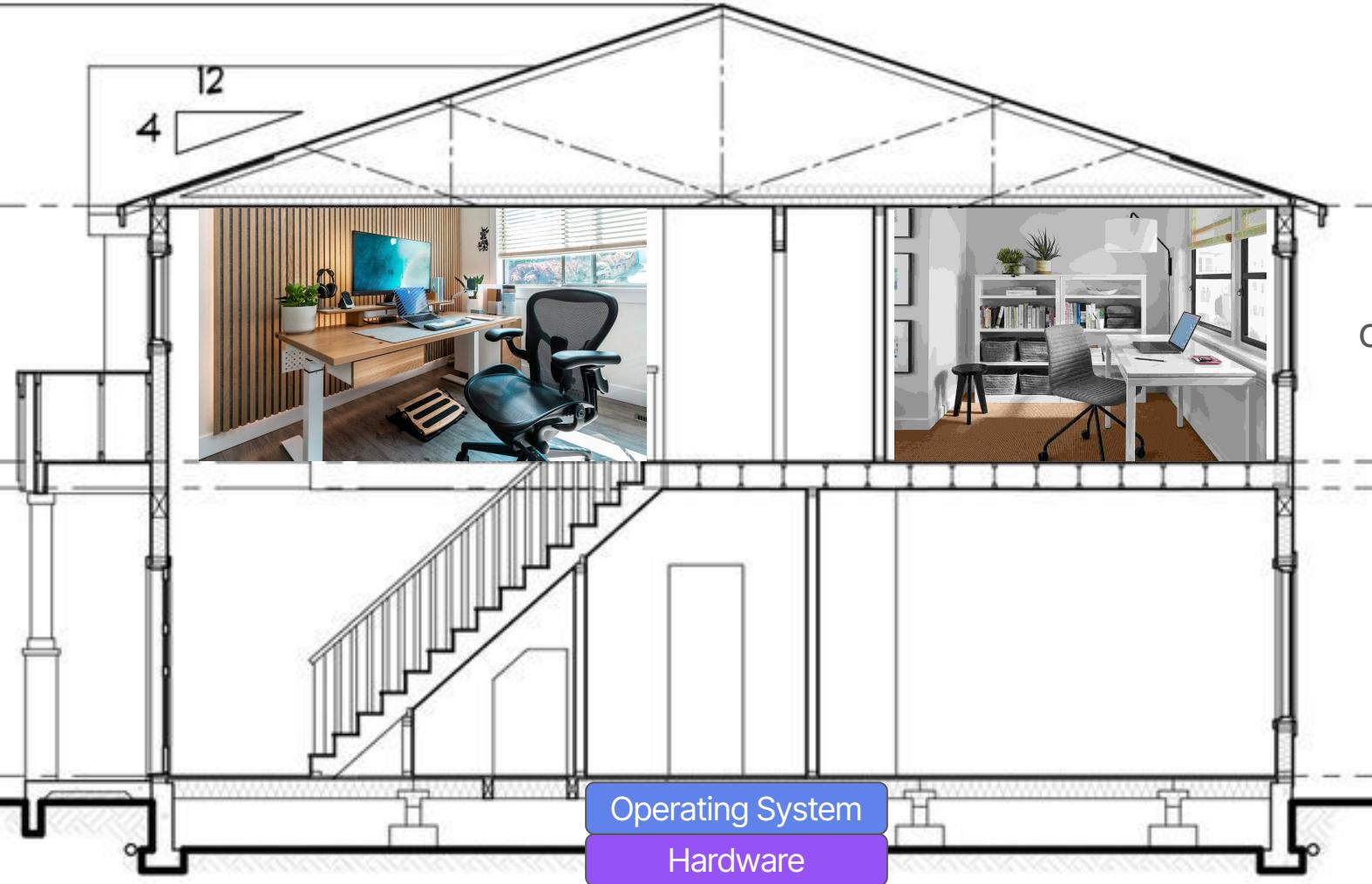
Containers





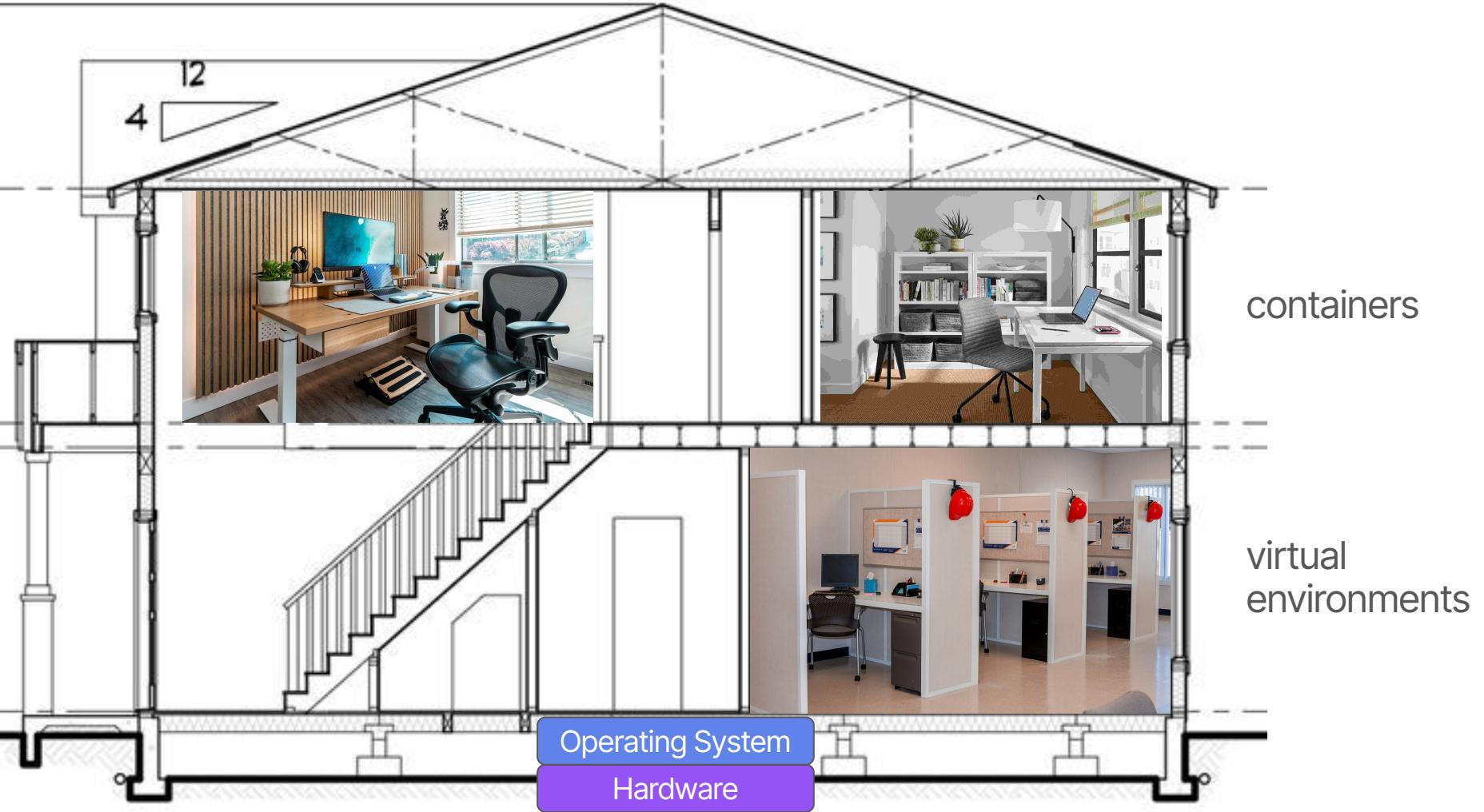
Operating System

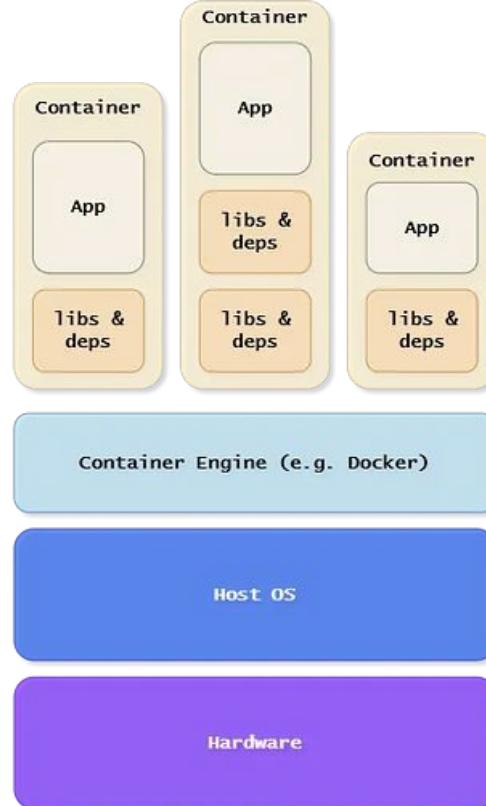
Hardware



Operating System

Hardware



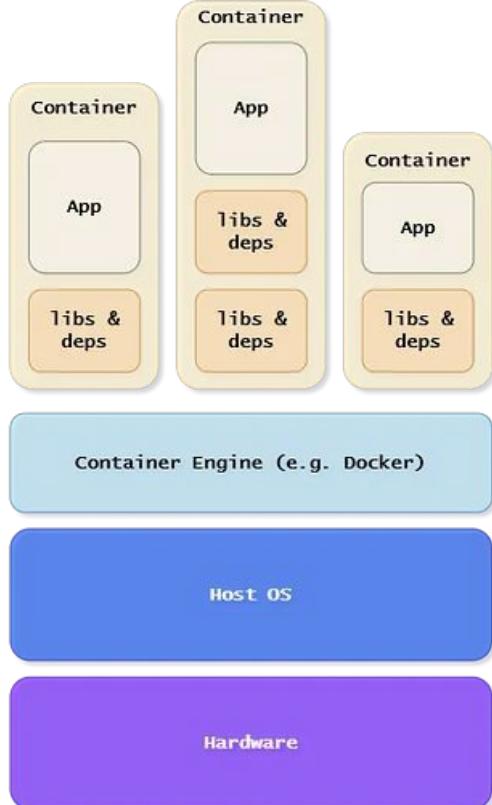


Bare Metal

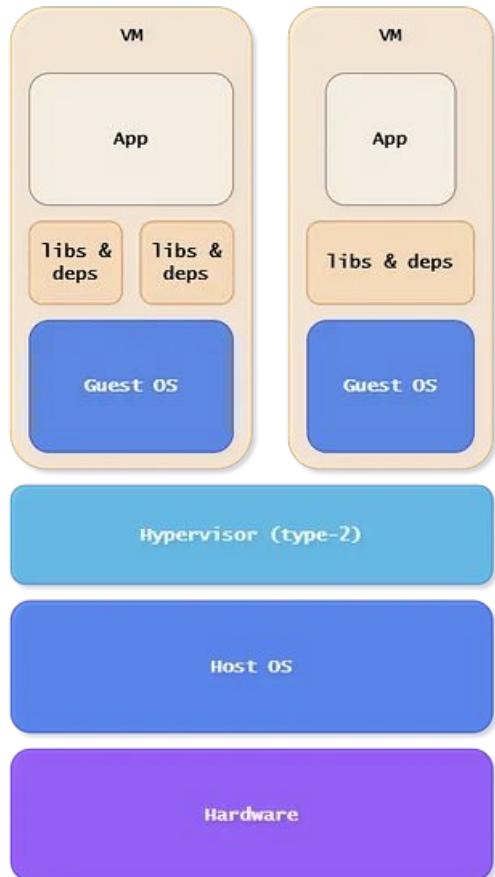
Containers



Bare Metal

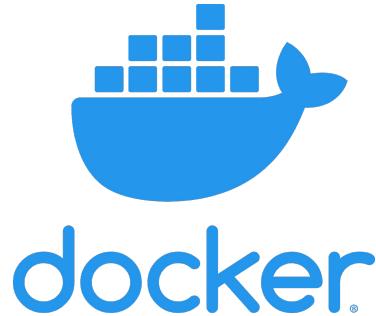


Containers



Virtual Machines

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

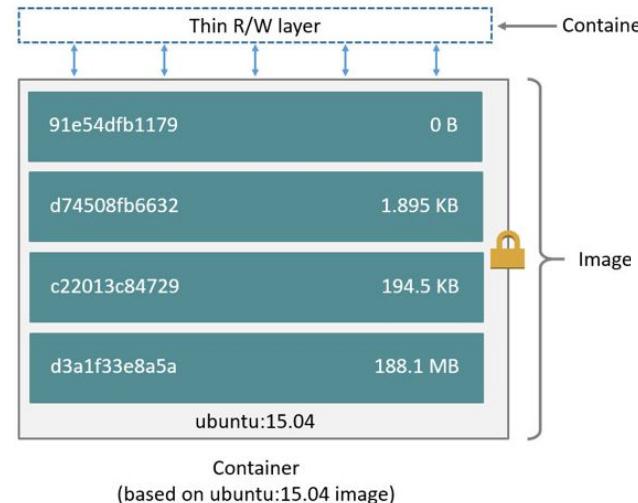
Docker **image** and **container**: what's the difference?

Docker image

- a **read-only** snapshot of an environment
- changing an image adds more layers
- can be stored on Docker Hub 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

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



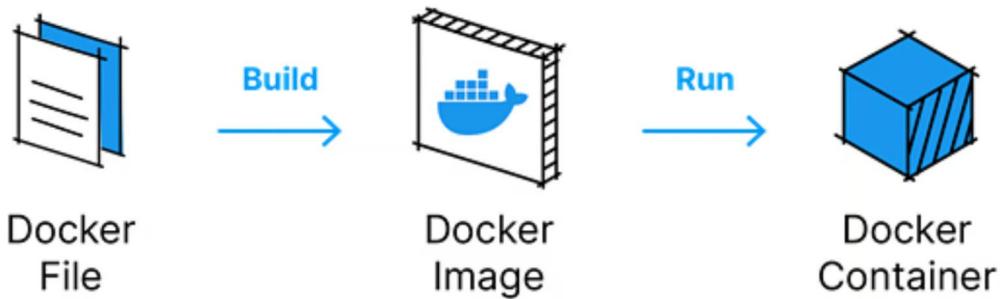
Build your **own** image

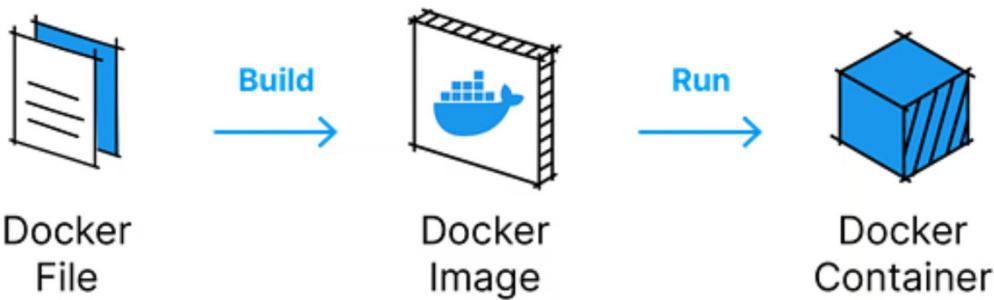
- a `Dockerfile` lets you describe the exact image you want to create
- Start from one image and add to it

```
# 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 file starts with a comment "# syntax=docker/dockerfile:1" followed by the instruction "FROM python:3.10-slim-buster". This line is highlighted with a red rectangular box and a red arrow points from the bottom-left towards it. The rest of the Dockerfile includes "WORKDIR", "COPY", "RUN" commands, and definitions for "ENTRYPOINT" and "CMD".





Exercise 1: Run a container from Docker Hub

- Find an image we like: https://hub.docker.com/_/hello-world
 - Already did this? Try <https://hub.docker.com/r/grycap/cowsay/>
- Pull the image (a specific tag)
- Run the image

Exercise 1: Run a container from Docker Hub

- Find an image we like: https://hub.docker.com/_/hello-world
 - Already did this? Try <https://hub.docker.com/r/grycap/cowsay/>
- Pull the image (a specific tag)
- Run the image



A screenshot of a terminal window titled 'surchs@deepthought:~/Documents/docker_place'. The window shows the following text:

```
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
```

 `docker --help` lists
main docker commands!

Exercise 1: Run a container from Docker Hub

Summary

- Docker Hub has pre-made images
- Image tags are important
- We can
 - retrieve images with `docker pull`
 - view images we have pulled with `docker images`
 - (pull 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

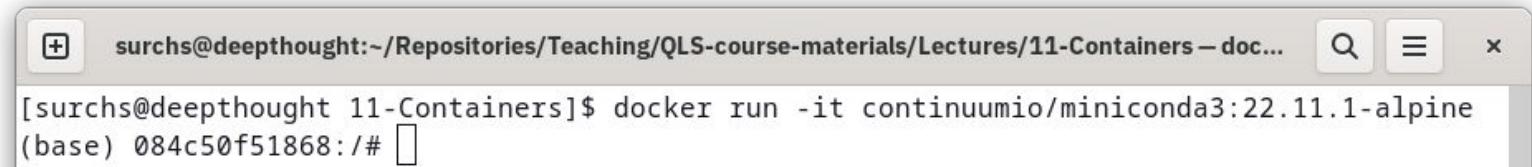
Exercise 2: Work with a container that has conda

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 main area shows the command: [surchs@deepthought 11-Containers]\$ docker run -it continuumio/miniconda3:22.11.1-alpine (base) 084c50f51868:/# . The 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 ~]$ hostname  
deepthought  
[surchs@deepthought ~]$ conda  
bash: conda: command not found...  
Install package 'conda' to provide command 'conda'? [N/y] ^C  
[surchs@deepthought ~]$  
[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 highlights the first two lines of the session, showing the user attempting to run 'conda'.
- A blue box highlights the line '(base) 8f67fd3f3c4c:/# hostname' and the resulting output '8f67fd3f3c4c'.
- A green box highlights the line '(base) 8f67fd3f3c4c:/# conda' and the usage information below it.

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).

inside
(in container)

outside (on host)

... and files written to a specific 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

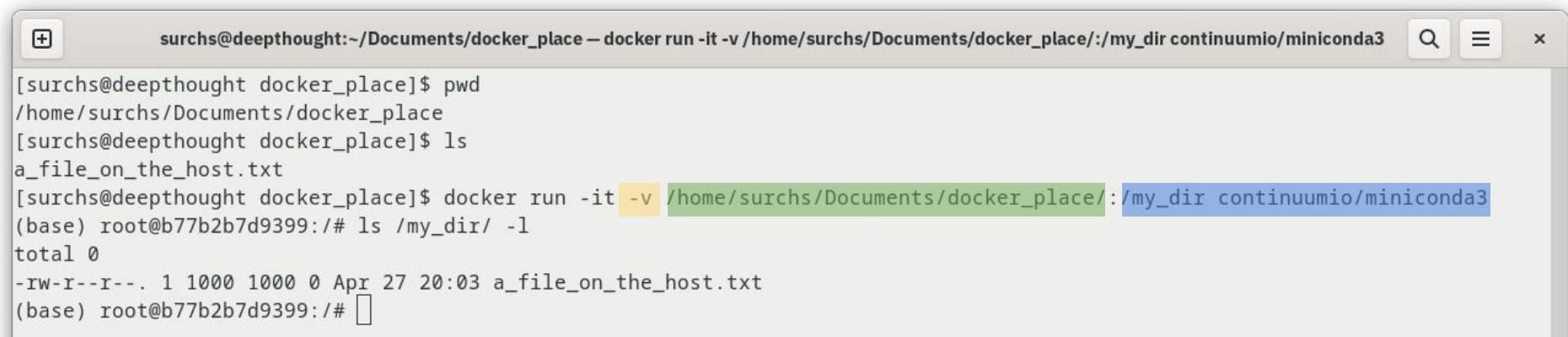
Don't write anything (worth keeping) to a container



docker ps -a
lists all containers

How do we share files between host and container?

- Bind mount a path on the host to a path in the container



The screenshot shows a terminal window with the following session:

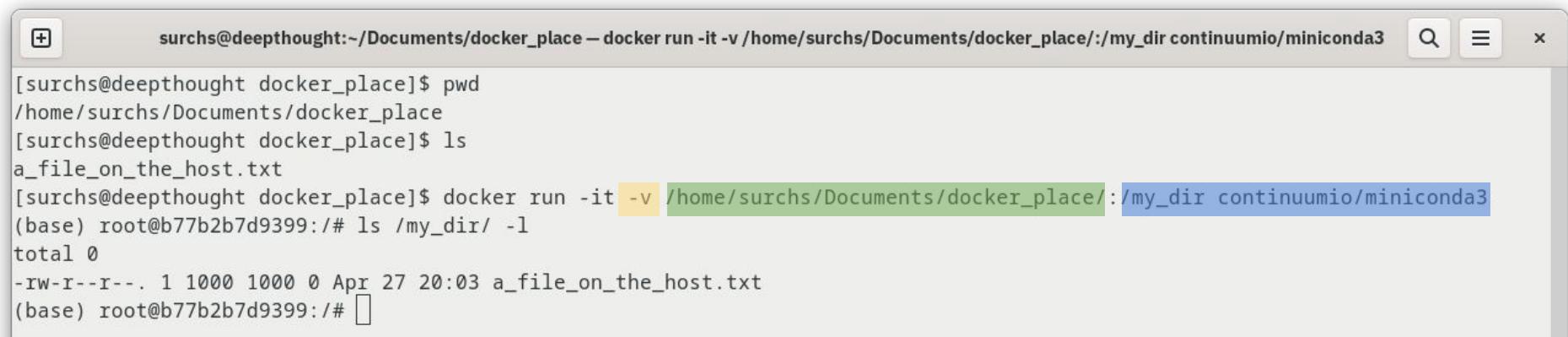
```
[surchs@deepthought docker_place]$ pwd  
/home/surchs/Documents/docker_place  
[surchs@deepthought docker_place]$ ls  
a_file_on_the_host.txt  
[surchs@deepthought docker_place]$ docker run -it -v /home/surchs/Documents/docker_place/:/my_dir continuumio/miniconda3  
(base) root@b77b2b7d9399:/# ls /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 command `-v /home/surchs/Documents/docker_place/:/my_dir` is highlighted in yellow, indicating it is the bind mount command.

- The target path will be created in the container, regardless of if it already exists
- Host path must start with / or ./ ! Or use `--mount` instead
- Container path must be an absolute path (start with /)

How do we share files between host and container?

- Bind mount a path on the host to a path in the container



The screenshot shows a terminal window with the following session:

```
[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 command `-v /home/surchs/Documents/docker_place/:/my_dir` is highlighted in yellow, indicating it is the bind mount configuration.

- The target path will be created in the container, regardless of if it already exists
- Host path must start with / or ./ ! Or use `--mount` instead
- Container path must be an absolute path (start with /)



which user owns files created by the container?

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 to 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
```

- This allows changes to persist beyond the container
- It's a bad idea to write directly into the container's filesystem

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 Data 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 [docker-options] <imagename> <command>`

Print the help text for the bids/validator:

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

Output:

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

...

Exercise 3: Run a Dockerized tool on local data

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

Print the help text for the bids/validator:

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

Output:

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

...

...How do I run other validator commands?

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

Exercise 3: Run a Dockerized tool on local data

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

Our test dataset is located on our **host machine** under:
~/QLS-course-materials/Lectures/2025/data/ds006

2. Run a bids/validator container again, but provide this dataset directory to validate.

Exercise 3: Run a Dockerized tool on local data

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

Our test dataset is located on our **host machine** under:
~/QLS-course-materials/Lectures/2025/data/ds006

2. Run a bids/validator container 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/on/host:/path/in/container`
- Where is the bids/validator app actually running? What files can it see?

Exercise 3: Run a Dockerized tool on local data

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

Our test dataset is located on our **host machine** under:
~/QLS-course-materials/Lectures/2025/data/ds006

2. Run a bids/validator container 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/on/host:/path/in/container`
- Where is the bids/validator app actually running? What files can it see?

```
cd ~/QLS-course-materials/Lectures/2025/data
docker run -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 it again on the dataset, but this time add **command options** to:

- Ignore warnings
- Ignore NIfTI header content

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 it again on the dataset, but this time add **command options** to:

- Ignore warnings
- Ignore NIfTI header content

HINTS:

- Print the help text again to see all command options for bids/validator
`docker run --rm bids/validator --help`

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 it again on the dataset, but this time add **command options** to:

- Ignore warnings
- Ignore NIfTI header content

HINTS:

- Print the help text again to see all command options for bids/validator
`docker run --rm bids/validator --help`

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

Apptainer (Singularity)



Apptainer is the container solution for HPCs (e.g. Compute Canada)

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

Instead, **Apptainer** (formerly Singularity) can be used to run containers:

Singularity^[1] 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*.



Singularity and Apptainer commands are nearly identical

Build images with Docker, run them with Apptainer

1. Pull an image from a Docker registry and create a local **Singularity Image File (.sif)**

```
$ apptainer pull docker://ghcr.io/apptainer/lolcow
```

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

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

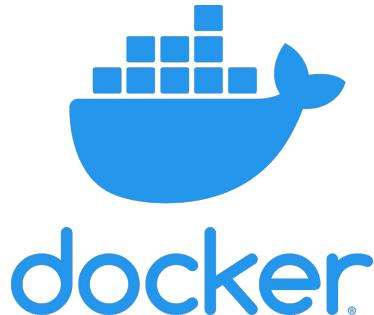
```
< Mon Aug 16 13:01:55 CDT 2021 >
```

```
-----  
 \ ^__^  
  \ oo)\_____  
   (__)\       )\/\  
    ||----w |  
    ||     ||
```

can also be Docker Hub

Containerization summary

- Docker **images** are **read-only snapshots**, you can find them on Docker Hub
- Images have tags (or versions), that you should specify when pulling
- A Docker **container** is an isolated, **live instance** of an image
- Docker **containers** have their own filesystem, but this is **not persistent**
- With **bind mounts** 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 minifies existing containers. Its purpose is to make it easier for scientists (and others) to easily create reproducible computational environments.

This website is in progress. Is there is something you would like to see here, please [submit a new GitHub issue](#).

```
neurodocker generate docker \
    --pkg-manager apt \
    --base-image neurodebian:bullseye \
    --ants version=2.4.3 \
    --miniconda version=latest conda_install="nipype notebook" \
    --user nonroot
```

Additional resources

- Docker docs <https://docs.docker.com/get-started/>
- Hands-on docker tutorials <https://training.play-with-docker.com/>,
<https://www.docker.com/101-tutorial/>
- 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>