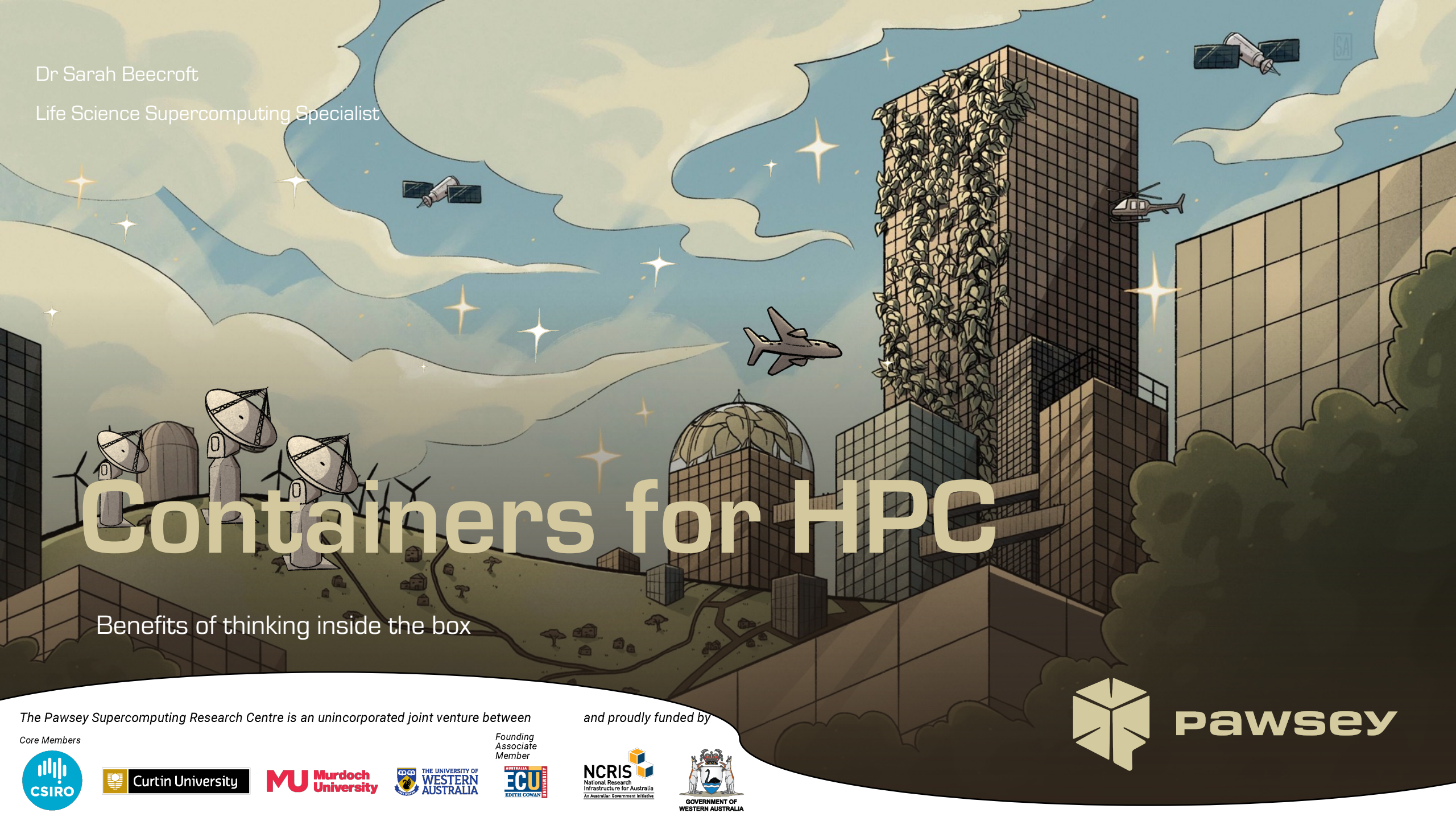
Dr Sarah Beecroft

Life Science Supercomputing Specialist



Containers for HPC

Benefits of thinking inside the box

The Pawsey Supercomputing Research Centre is an unincorporated joint venture between

Core Members

CSIRO

Curtin University

Murdoch University

The University of Western Australia

Founding Associate Member

ECU Edith Cowan University

and proudly funded by

NCRIS National Research Infrastructure for Australia An Australian Government Initiative

Government of Western Australia

Pawsey

Setonix named the fourth greenest supercomputer in the world

X30

more powerful than our former HPC systems

42*

petaFLOPS of power, making it the fastest research supercomputer in the Southern Hemisphere

Advanced architecture

Nimbus (Cloud)

100 Gbps

Setonix Supercomputer

100 Gbps

ASKAP Ingest

10 Gbps

Garrawarla

10 Gbps

100 Gbps

100 Gbps
External Firewall

S3 Interface

Fibre Channel

70 PB tape system

100Gbps

85+ PB Ceph Object Store

Overview of Containers

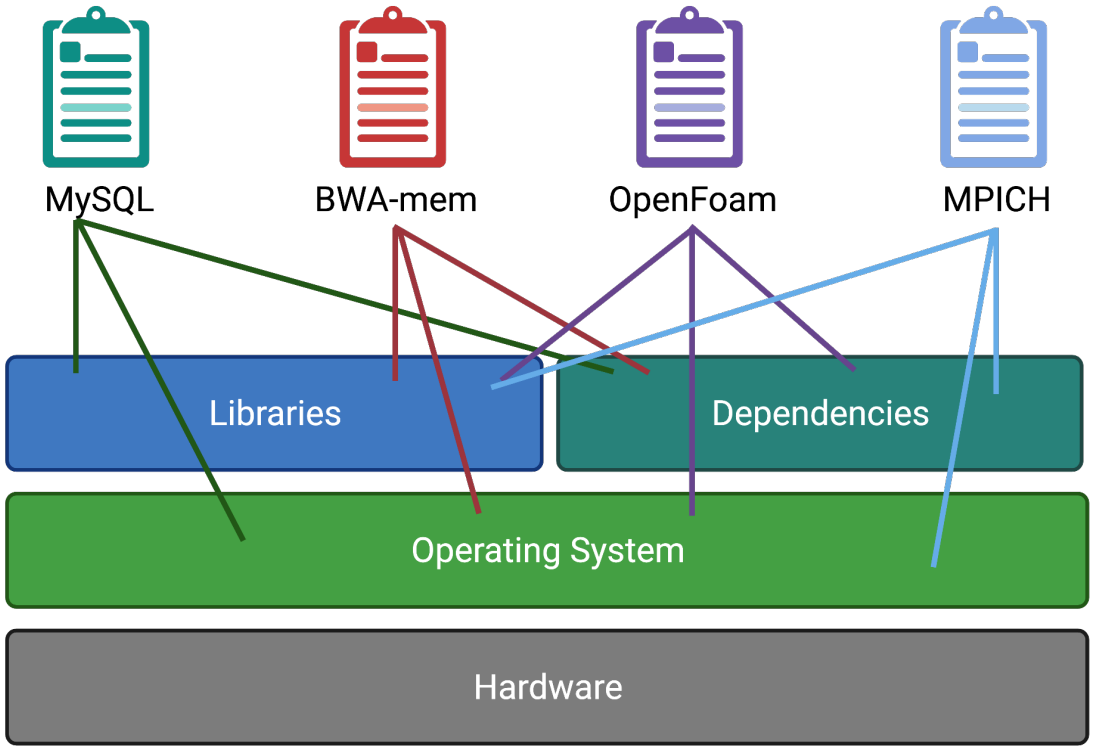
What can you use containers for?

- Avoid local installation of packages (e.g. Conda/Pip/apt)
- Handling clashing/multiple versions
- Shipping software you've published
- Fitting in with NextFlow/Snakemake/AirFlow
- Bleeding edge versions that wouldn't work bare metal

Good Use Cases for Containers

- Large volume of software to be installed (e.g. bioinformatics)
- Complex builds with lots of dependencies (e.g., deep learning frameworks, bioinformtics)
- Dependency trees with tens/hundreds of small packages (e.g. Python)
    - Single container file can improve performance on parallel filesystems
- Intensive I/O patterns in software that write a large number of small files (e.g. OpenFoam)
    - Single container file can improve performance on parallel filesystems

How do programs usually run?

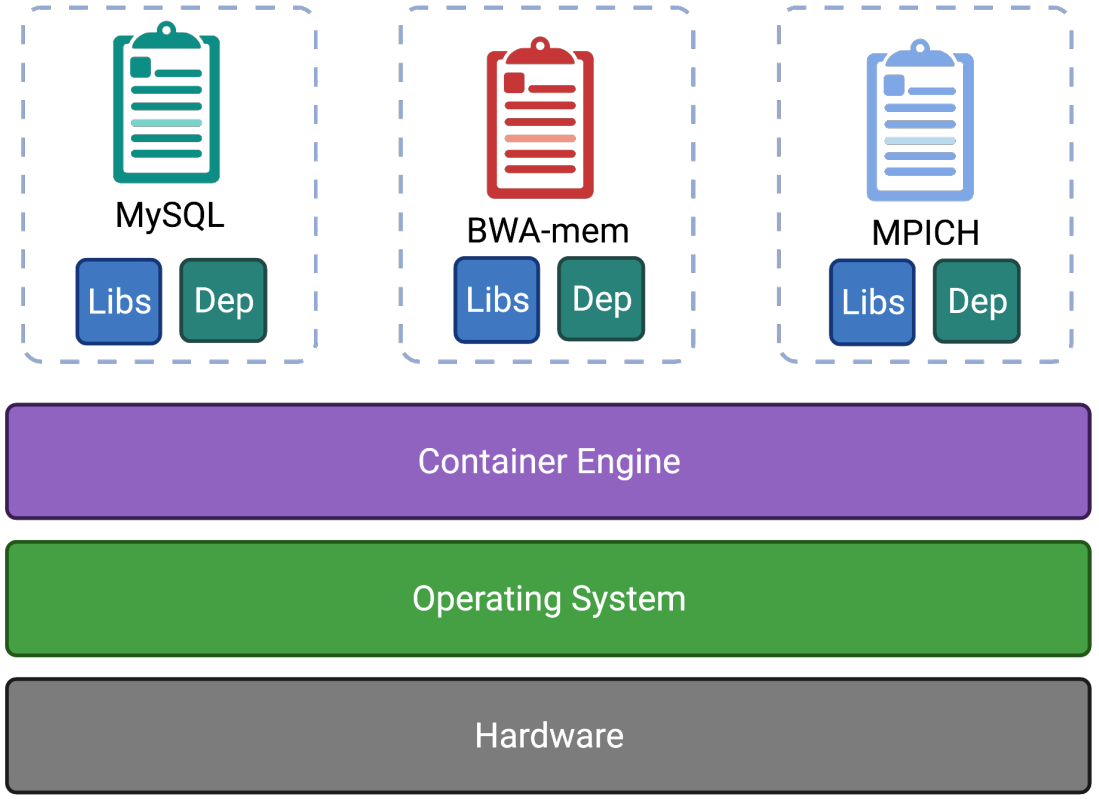


Programs rely on shared libraries/dependencies

These may clash or be incompatible

AKA: Dependency Hell

How do programs run with containers?



- Self-contained environments
- Encapsulates everything you need for a software to run
- Runs independently from host machine (e.g. your laptop/cluster)

Uses a combination of Kernel “cgroups” and “namespaces” to create isolated environments

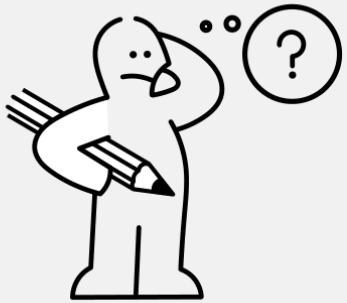
Containers encapsulate everything for a software to run

Why containers?

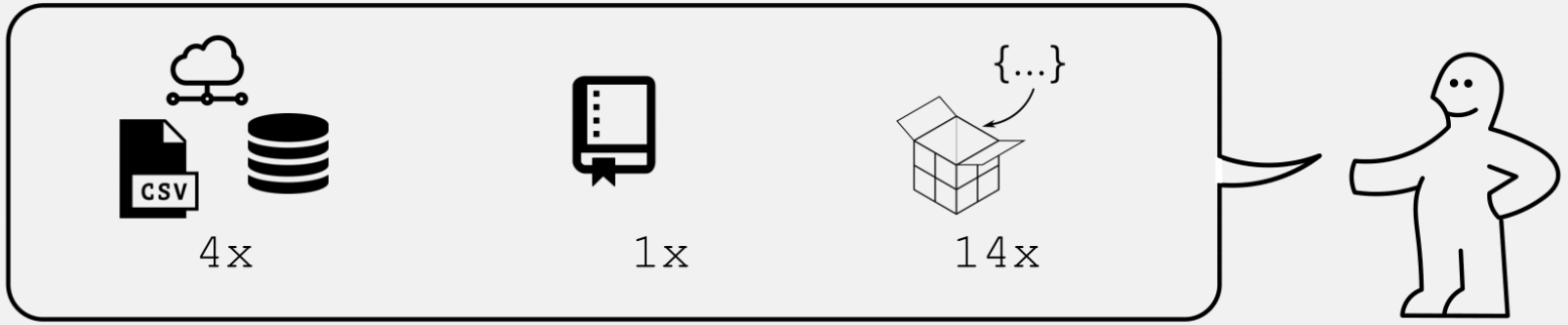
Enabling workflows that are:

- Reproducible
- Robust
- Portable
- Easier to maintain

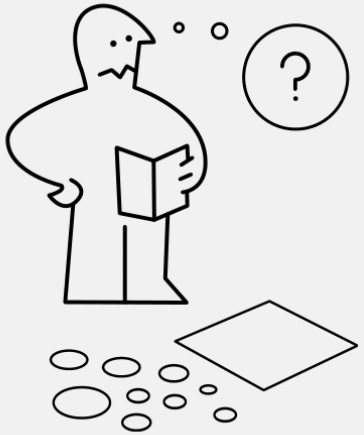
1.



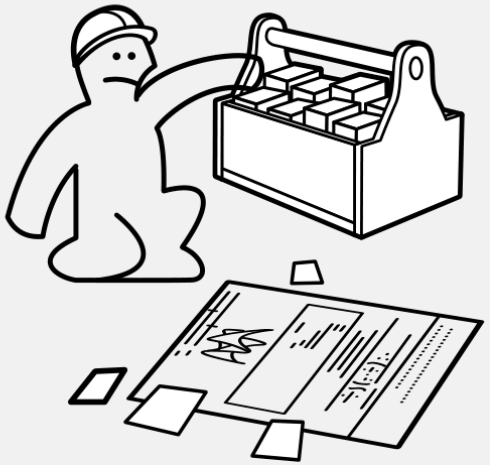
2.



3.

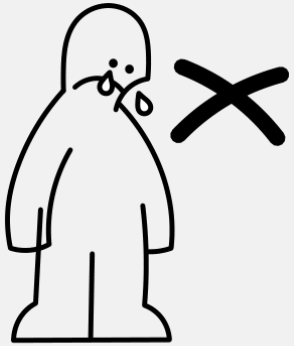


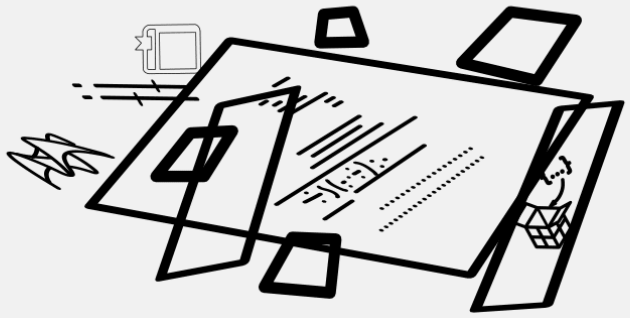
4.



5.

6.





- Reproducible:
  - Repeated analysis == identical results
- Robust:
  - Identical tool versions over time
  - Container won't suddenly break

Image: https://github.com/karthik/rstudio2019

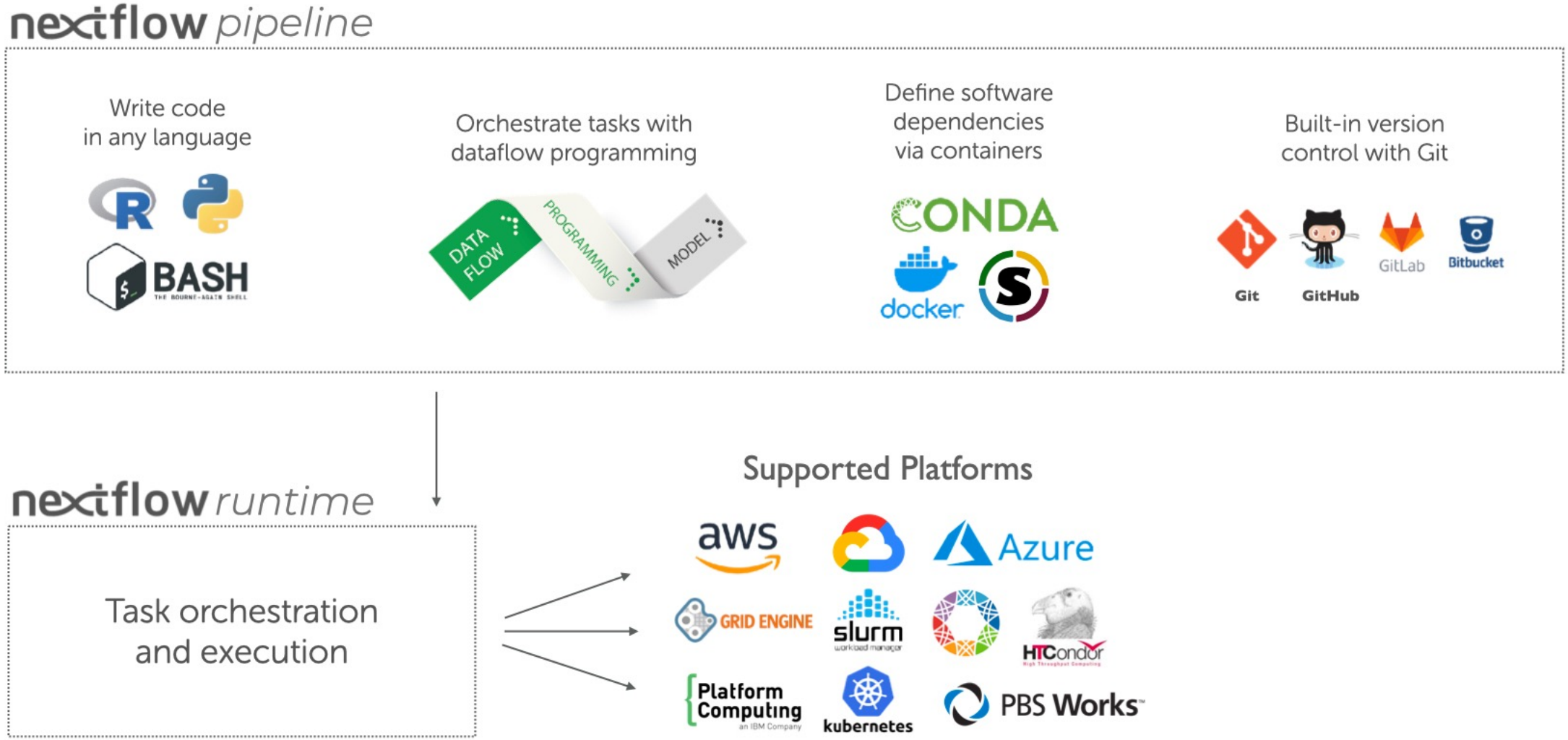
Portability

- Easy to send your exact computing environment to any system
- Saves time when moving workflow to new machine(s)
- Enables collaboration
- Increasingly important for scientific publishing

Easier to Maintain

- Write one install recipe, use it everywhere
- Solve dependency hell once and for all
- More time for other work!

Containers support workflow engines



Container Details

Some container keywords

Recipe

Instructions for making a container (image)

Build

Act of “compiling” the recipe into an image

Image

“Compiled” container that’s not currently running. Gets shipped around

Repository

Place for images to be stored and shared

Container

Running instantiation of an image

Container engine

Software that allows the container to come alive and run

Container keywords

Recipe (aka Dockerfile)

pawsey-containers / pytorch / **pytorch.dockerfile**

dipietrantonio Adds the PyTorch dockerfile.

Code | Blame | 39 lines (35 loc) · 1.14 KB

```
1  FROM quay.io/pawsey/rocm-mpich-base:rocm5.6.0-mpich3.4.3-ubuntu22
2
3  ENV _GLIBCXX_USE_CXX11_ABI=1
4  ENV USE_CUDA=0
5  ENV USE_ROCM=1
6  ENV CXX=g++
7  ENV CC=gcc
8  ENV CXXFLAGS=-std=c++17
9  ENV PYTORCH_ROCM_ARCH=gfx90a
10
11 RUN     apt -y install libopenblas-dev \
12         && (! [ -e /tmp/build ] || rm -rf /tmp/build) \
13         && mkdir /tmp/build && cd /tmp/build \
14         # install eigen
15         && wget https://gitlab.com/libeigen/eigen/-/archive/3.4.0/eigen-3.4.0.tar.gz \
16         && tar xf eigen-3.4.0.tar.gz \
17         && cd eigen-3.4.0 \
18         && mkdir build \
19         && cd build \
20         && cmake .. \
21         && make -j 16 \
22         && make install
23
24 RUN     cd /tmp/build \
25         && git clone --branch v2.1.0 --recursive https://github.com/pytorch/pytorch \
26         && cd pytorch \
27         && grep -R . -e "MPI_CXX" | cut -f1 -d: | xargs -n1 sed -i -e "s/MPI_CXX/MPI_C/g" \
28         # if you are updating an existing checkout \
29         && git submodule sync \
30         && git submodule update --init --recursive \
31         # sets "USE_SYSTEM_EIGEN_INSTALL=ON"
32         && sed -i -e '270d' -e '269a ON)' CMakeLists.txt \
33         # Install deps
34         && python3 -m pip install -r requirements.txt \
35         && make triton\
36         && python3 tools/amd_build/build_amd.py\
37         && python3 setup.py install
38
39 RUN     [ -e /tmp/build ] && rm -rf /tmp/build
```

Build

Image(s) hosted on a container **repository**

dockerhub | pytorch | Sign In | Sign up

Explore / pytorch/pytorch

pytorch/pytorch Sponsored OSS ☆1.1K

Pulls 10M+

By PyTorch · Updated about 1 month ago

PyTorch is a deep learning framework that puts Python first.

IMAGE

DATA SCIENCE | LANGUAGES & FRAMEWORKS | MACHINE LEARNING & AI

Overview | Tags

Sort by Newest | Filter Tags

Tags indicate version and other info

TAG
2.3.1-cuda11.8-cudnn8-devel
Last pushed a month ago by pytorchbot

```
docker pull pytorch/pytorch:2.3.1-cuda11.8-cudnn8-devel
```
Copy

Digest	OS/ARCH	Compressed Size
5c2e1d0bbe0f	linux/amd64	8.73 GB

TAG
2.3.1-cuda11.8-cudnn8-runtime
Last pushed a month ago by pytorchbot

```
docker pull pytorch/pytorch:2.3.1-cuda11.8-cudnn8-runtime
```
Copy

Digest	OS/ARCH	Compressed Size
ff97981d417f	linux/amd64	3.76 GB

TAG
2.3.1-cuda12.1-cudnn8-devel
Last pushed a month ago by pytorchbot

```
docker pull pytorch/pytorch:2.3.1-cuda12.1-cudnn8-devel
```
Copy

What is a container engine?

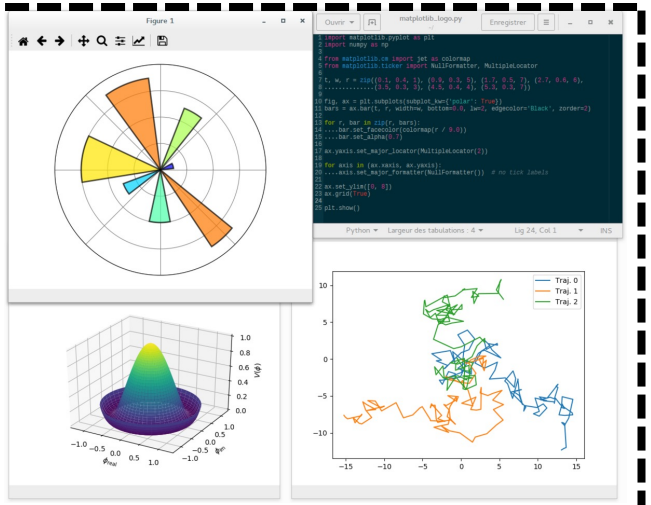
User

Command

docker

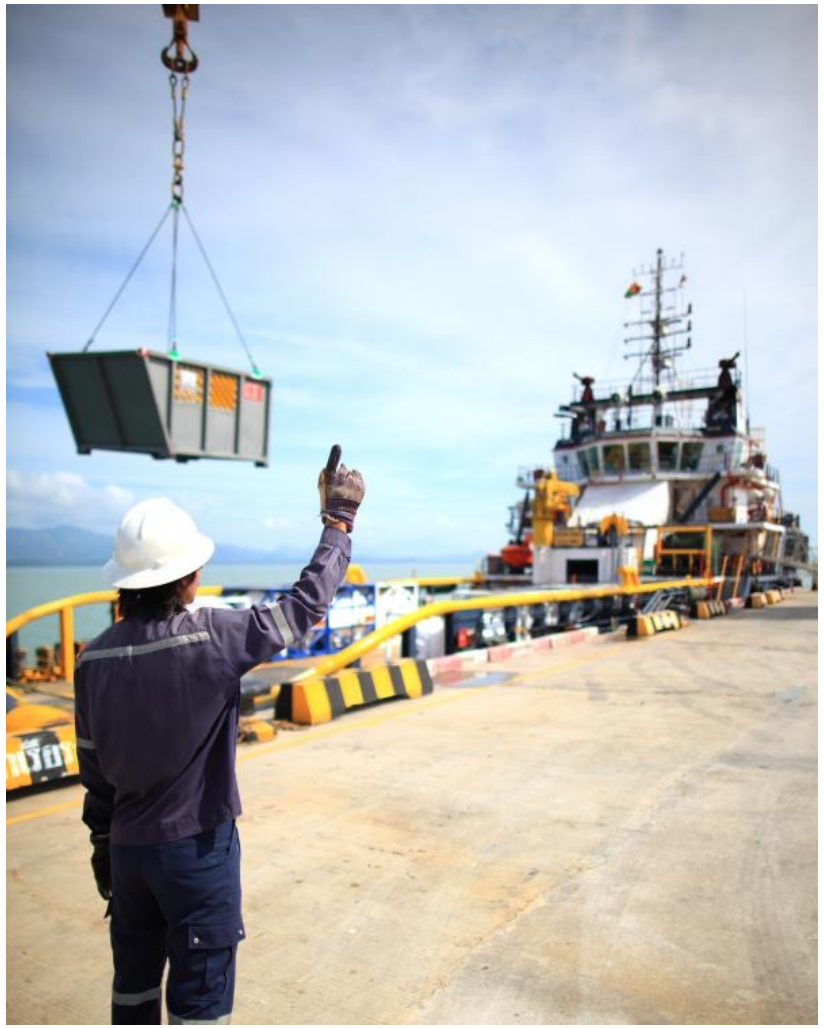
Container engine

Execution



Container runs task

Popular container engines



Singularity

SHIFTER

Charliecloud

docker

Container Runtime

- Container file-system is read-only
- Filesystem in a file (squashfs/SIF) is performant on parallel file-systems
- User-friendly features
  - Same user as in host
  - Same working directory as in host by default
  - Shell variables inherited from host

Using Singularity: key commands

Key commands

- Download: `singularity pull docker://[registry/][repository/]name:tag`
- Execute commands: `singularity exec container_file.sif command options`
- Open shell inside container: `singularity shell container_file.sif`

- How to run GPU-enabled containers
  - Additional Singularity option
    - Nvidia GPUs: `singularity exec --nv container_file.sif command options`
    - AMD GPUs: `singularity exec --rocm container_file.sif command options`

Example: Download and Run Container with Singularity

```
$ python3 --version
Python 3.6.13

$ cd $MYSCRATCH
$ module load singularity/3.8.6
$ singularity pull docker://python:3.9.13-slim
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
..
INFO:    Creating SIF file...

$ singularity exec python_3.9.13-slim.sif python3 --version
Python 3.9.13
```

Container runtime – Singularity

Isolate the environment: **`-e`** or **`--cleanenv`**

Pass specific environment variables

- **`export SINGULARITYENV_VARIABLE=value`**
- [Singularity 3.6.x on] **`--env VARIABLE=value`**

More options: **`singularity exec --help`**

Do I need to always create new container recipes?

- Container registries have 1000s of ready-to-deploy containers
- Can use ‘off the shelf’, or edit existing recipes

A recipe puts Docker specific keywords around your normal installation instructions

```
# Example container recipe

FROM <image>

RUN ..installation commands..
..

ENV ..define shell variables..
..

ADD/COPY ..copy files from host to image..
```

pawsey-containers / pytorch / pytorch.dockerfile

dipietrantonio Adds the PyTorch dockerfile.

Code | Blame | 39 lines (35 loc) · 1.14 KB

```
1   FROM quay.io/pawsey/rocm-mpich-base:rocm5.6.0-mpich3.4.3-ubuntu22
2
3   ENV _GLIBCXX_USE_CXX11_ABI=1
4   ENV USE_CUDA=0
5   ENV USE_ROCM=1
6   ENV CXX=g++
7   ENV CC=gcc
8   ENV CXXFLAGS=-std=c++17
9   ENV PYTORCH_ROCM_ARCH=gfx90a
10
11  RUN     apt -y install libopenblas-dev \
12          && (! [ -e /tmp/build ] || rm -rf /tmp/build) \
13          && mkdir /tmp/build && cd /tmp/build \
14          # install eigen
15          && wget https://gitlab.com/libeigen/eigen/-/archive/3.4.0/eigen-3.4.0.tar.gz \
16          && tar xf eigen-3.4.0.tar.gz \
17          && cd eigen-3.4.0 \
18          && mkdir build \
19          && cd build \
20          && cmake .. \
21          && make -j 16 \
22          && make install
23
24  RUN     cd /tmp/build \
25          && git clone --branch v2.1.0 --recursive https://github.com/pytorch/pytorch \
26          && cd pytorch \
27          && grep -R . -e "MPI_CXX" | cut -f1 -d: | xargs -n1 sed -i -e "s/MPI_CXX/MPI_C/g" \
28          # if you are updating an existing checkout \
29          && git submodule sync \
30          && git submodule update --init --recursive \
31          # sets "USE_SYSTEM_EIGEN_INSTALL=ON"
32          && sed -i -e '270d' -e '269a ON)' CMakeLists.txt \
33          # Install deps
34          && python3 -m pip install -r requirements.txt \
35          && make triton\
36          && python3 tools/amd_build/build_amd.py\
37          && python3 setup.py install
38
39  RUN     [ -e /tmp/build ] && rm -rf /tmp/build
```

Can I share files with the host machine?

“Bind-mounting” allows you to share directories between the container and the host machine

```
# Bind mount flag example


--bind dir1,dir2,dir3
```

Real example:

```
singularity exec -B /scratch/user1/inputs:/usr/local/interproscan/data -e
interproscan_latest.sif interproscan.sh <flags>
```

Tidying up your filesystem with persistent overlays

- Too many small files stress the parallel file system
- Singularity overlays create a virtual filesystem that holds many files but appears as one file to the host system

Example overlay creation

Create on host system location for overlay

```
> mkdir -p overlay/upper
```

Create overlay of 500MB of empty space and include above directory

```
> dd if=/dev/zero of=overlay.img bs=1M count=500

> mkfs.ext3 -d overlay overlay.img
```

Container is writable as the unprivileged user who created the overlay/upper directory

```
> singularity shell --overlay overlay.img ubuntu.sif
```

Best Practices

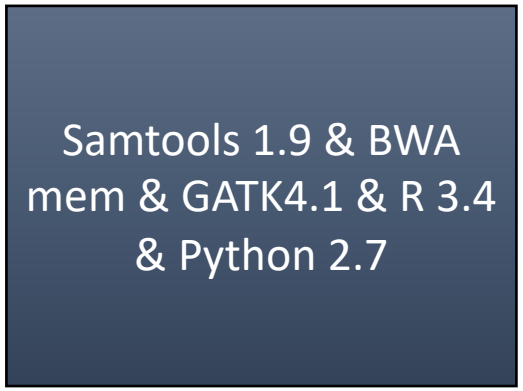
Should I put everything into one container?

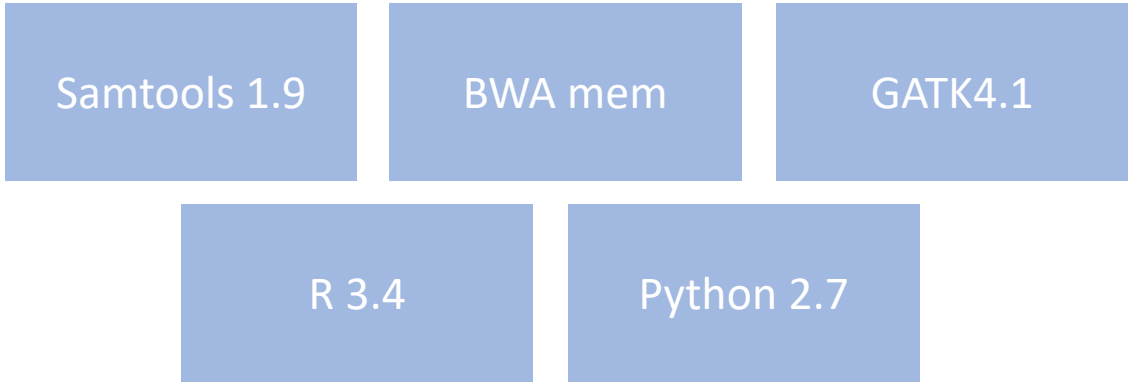
Monolithic container

- Holds many independent programs
- Must re-build to update one thing
- Edging towards dependency hell

Modular containers

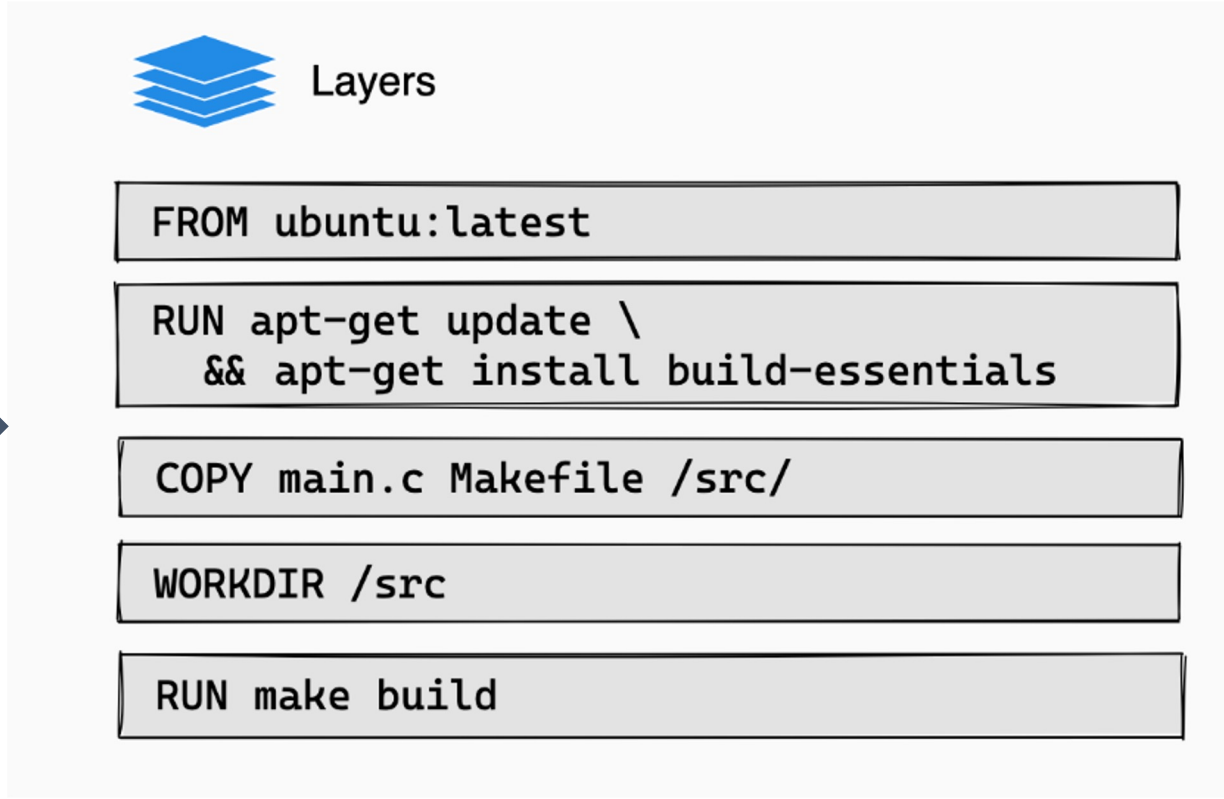
- Swap components as needed
- Keep tools isolated
- Only edit one component as needed





Each instruction in your recipe becomes a layer in the container

```
# syntax=docker/dockerfile:1

FROM ubuntu:latest


RUN apt-get update && apt-get
install -y build-essentials

COPY main.c Makefile /src/

WORKDIR /src/

RUN make build
```



Best Practice – Combine RUN commands to reduce layer size

```
RUN wget http://hostname.com/mycode.tgz
RUN tar xzf mycode.tgz
RUN cd mycode ; make; make install
RUN rm -rf mycode.tgz mycode
```

- *Solution:*

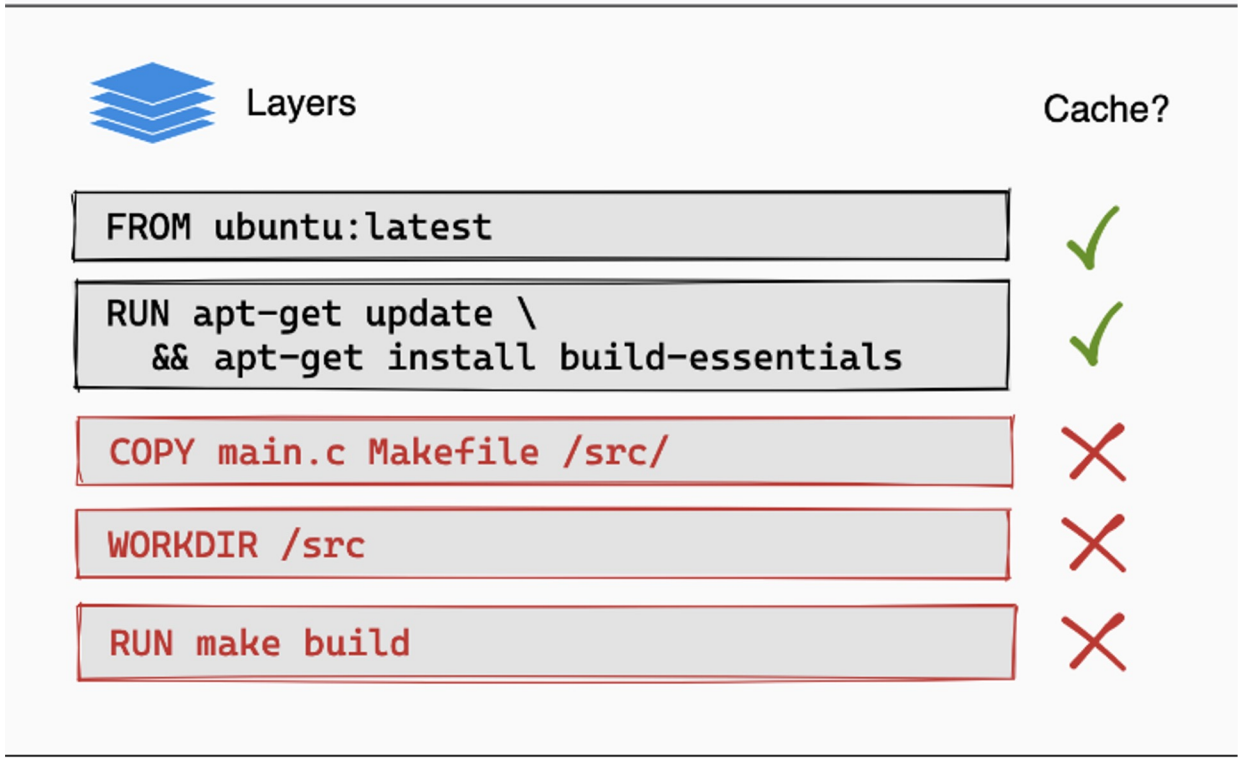
```
RUN wget http://hostname.com/mycode.tgz && \
   tar xzf mycode.tgz && \
   cd mycode && make && make install && \
   rm -rf mycode.tgz mycode
```

Note: Docker doesn't clean up *anything* so think about what you need to purge to keep your image small

Best Practice – Order matters, exploit the build cache

- Expensive steps → near the beginning of the Dockerfile

- Frequently changing steps → near the end of the Dockerfile
  - Avoid triggering rebuilds of layers that haven't changed.



Best Practice – Use version tags to manage base images

```
FROM python:latest # This can change over time
```

- *Solution:*

```
FROM python:3.7 # This will remain a 3.7 version
```

- Create a base image that contains dependencies that shouldn't need to change frequently
- Tag that base image with a version
- Use this as the base image for the application or other components that change more frequently
- This makes it easier to freeze all of the dependencies and avoid accidental updates

Example: PyTorch

pawsey-containers / mpi / rocm-mpich-base / **buildrocm-mpich-base.dockerfile**

Ubuntu fixed additional command necessary prior to rocm PR

Code | Blame | 270 lines (253 loc) · 11.7 KB

```
ARG OS_VERSION="22.04"
FROM ubuntu:${OS_VERSION}
# redefine after FROM to ensure it is defined
ARG OS_VERSION="22.04"

#libfabric version
ARG LIBFABRIC_VERSION=1.18.1
# mpich version
ARG MPICH_VERSION="3.4.3"
# lustre version
ARG LUSTRE_VERSION="2.15.0-RC4"
# mpi4py version
ARG MPI4PY_VERSION="3.1.4"


#define some metadata
LABEL org.opencontainers.image.created="2024-02"
LABEL org.opencontainers.image.authors="Cristian Di Pietratonio <cristian.dipietrantonio@pawsey.org.au>, Pascal Elahi <pascal.elahi@pawsey.org.au>"
LABEL org.opencontainers.image.documentation="https://github.com/PawseySC/pawsey-containers/"
LABEL org.opencontainers.image.source="https://github.com/PawseySC/pawsey-containers/mpi/mpich-base/buildmpich.dockerfile"
LABEL org.opencontainers.image.vendor="Pawsey Supercomputing Research Centre"
LABEL org.opencontainers.image.licenses="GNU GPL3.0"
LABEL org.opencontainers.image.title="Setonix compatible MPICH + ROCM base"
LABEL org.opencontainers.image.description="Common base image providing mpi + rocm compatible with cray-mpich used on Setonix"
LABEL org.opencontainers.image.base.name="pawsey/mpibase:ubuntu${OS_VERSION}-mpich-${MPICH_VERSION}.setonix"

# Install required packages and dependencies
ARG LINUX_KERNEL=5.15.0-91
# for newer ubuntu might want to use newer kernels like 6.2.0-39
ENV DEBIAN_FRONTEND="noninteractive"
RUN echo "Install apt packages" \
    && apt-get update -qq \
    && apt-get -y --no-install-recommends install \
        build-essential \
        gnupg gnupg2 \
        ca-certificates \
        gdb \
        gcc-12 g++-12 gfortran-12 \
        wget \
        git \
        python3-six python3-setuptools \
        patchelf strace ltrace \
```

pawsey-containers / pytorch / **pytorch.dockerfile**

dipietrantonio Adds the PyTorch dockerfile.

Code | Blame | 39 lines (35 loc) · 1.14 KB

```
FROM quay.io/pawsey/rocm-mpich-base:rocm5.6.0-mpich3.4.3-ubuntu22

ENV _GLIBCXX_USE_CXX11_ABI=1
ENV USE_CUDA=0
ENV USE_ROCM=1
ENV CXX=g++
ENV CC=gcc
ENV CXXFLAGS=-std=c++17
ENV PYTORCH_ROCM_ARCH=gfx90a

RUN     apt -y install libopenblas-dev \
        && (! [ -e /tmp/build ] || rm -rf /tmp/build) \
        && mkdir /tmp/build && cd /tmp/build \
        # install eigen
        && wget https://gitlab.com/libeigen/eigen/-/archive/3.4.0/eigen-3.4.0.tar.gz \
        && tar xf eigen-3.4.0.tar.gz \
        && cd eigen-3.4.0 \
        && mkdir build \
        && cd build \
        && cmake .. \
        && make -j 16 \
        && make install

RUN     cd /tmp/build \
        && git clone --branch v2.1.0 --recursive https://github.com/pytorch/pytorch \
        && cd pytorch \
        && grep -R . -e "MPI_CXX" | cut -f1 -d: | xargs -n1 sed -i -e "s/MPI_CXX/MPI_C/g" \
        # if you are updating an existing checkout \
        && git submodule sync \
        && git submodule update --init --recursive \
        # sets "USE_SYSTEM_EIGEN_INSTALL=ON"
        && sed -i -e '270d' -e '269a ON)' CMakeLists.txt \
        # Install deps
        && python3 -m pip install -r requirements.txt \
        && make triton\
        && python3 tools/amd_build/build_amd.py\
        && python3 setup.py install

RUN     [ -e /tmp/build ] && rm -rf /tmp/build
```

Best Practice – Use Trusted images

```
FROM foobar/python:3.7 # do you know foobar?
```

- *Solution:*

```
FROM python:3.7 # official image from Python Foundation
```

Best Practice – Use versioned dependencies

```
RUN git clone https://github.com/foo/bar.git
RUN cd bar && make install
```

- *Solution: (if you have a tagged release)*

```
RUN git clone --branch v1.0.3 --depth 1 https://github.com/foo/bar.git
RUN cd bar && make install
```

- *Solution: (if you have a commit hash)*

```
RUN git clone https://github.com/foo/bar.git
RUN cd bar && git checkout 4e3c9cc && make install
```

Best Practice – Avoid Semicolons; Use Ampersands &&

```
RUN wget http://hostname.com/mycode.tgz ; \
   tar xzf mycode.tgz ; \
   cd mycode ; make ; make install ; \
   rm -rf mycode.tgz mycode
```

- *Solution:*

```
RUN wget http://hostname.com/mycode.tgz && \
   tar xzf mycode.tgz && \
   cd mycode && make && make install && \
   rm -rf mycode.tgz mycode
```

Pawsey.org.au

sarah.beecroft@csiro.au

Any questions?

Thank you

Conda containers pro-tips: Make your build-time faster

- Mamba is a faster drop-in replacement for conda. Can use **`conda-forge/mambaforge3`** base image
  - Otherwise **`continuumio/miniconda3`** is a good option

- Optional: Avoid updating existing packages with **`--freeze-installed`** if doing multiple rounds of installation

Acknowledgement: https://uwekorn.com/2021/03/01/deploying-conda-environments-in-docker-how-to-do-it-right.html

Conda containers pro-tips: Make your conda env portable across O/S

```
conda env export --from-history
```

```
dependencies:
  ...
  - libcurl=7.71.1=h9bf37e3_8
  - libcxx=11.0.1=habf9029_0
  - libedit=3.1.20191231=hed1e85f_2
  ...
```

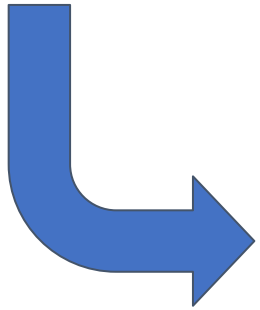
```
dependencies:
  ...
  - libcurl=7.71.1
  - libcxx=11.0.1
  - libedit=3.1.20191231
  ...
```

Conda containers pro-tips: Make you build-time faster

Avoid re-downloading unchanged packages with Docker cache

- The cache will persist between runs and will be shared between concurrent builds
- Ensure Buildkit is enabled (**`export DOCKER_BUILDKIT=1`**)

```
RUN conda env create -f env.yml
```



```
RUN --mount=type=cache,target=/opt/conda/pkgs \
    conda env create -f env.yml
```