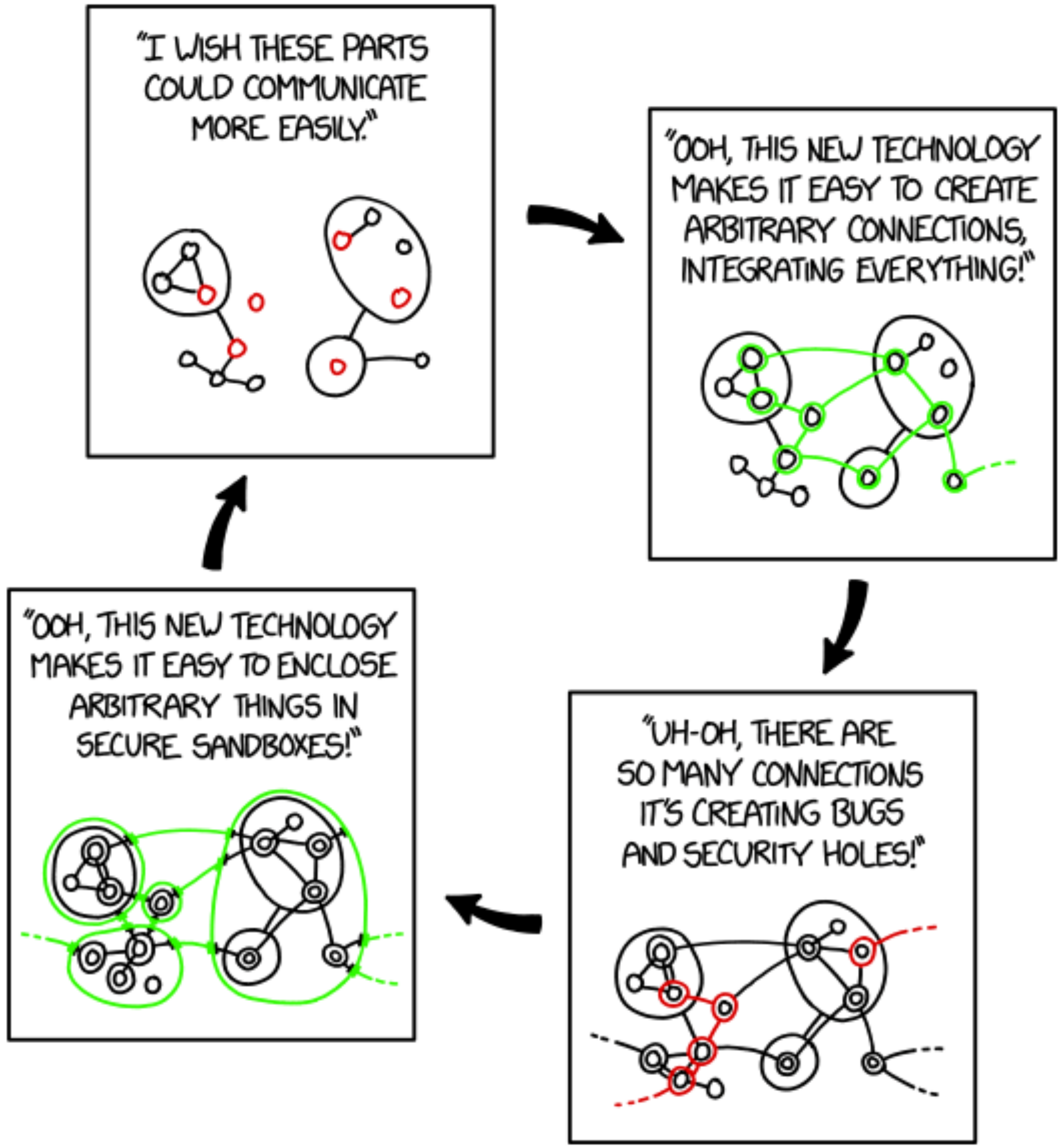
Containers

Joseph Long
Code Coffee • 2019-01-22

Photo by chuttersnap on Unsplash


"I WISH THESE PARTS COULD COMMUNICATE MORE EASILY."
"OOH, THIS NEW TECHNOLOGY MAKES IT EASY TO CREATE ARBITRARY CONNECTIONS, INTEGRATING EVERYTHING!"
"UH-OH, THERE ARE SO MANY CONNECTIONS IT'S CREATING BUGS AND SECURITY HOLES!"
"OOH, THIS NEW TECHNOLOGY MAKES IT EASY TO ENCLOSE ARBITRARY THINGS IN SECURE SANDBOXES!"

"Sandboxing Cycle" by Randall Munroe (xkcd) CC-BY-NC 2.5

Ideal world

```
$ pip install myutility
$ myutility
Welcome to myutility!
```

Real world

```
$ pip install myutility
$ myutility
myutility depends on ffmpeg, please install
it

$ apt-get install ffmpeg
Error: permission denied

$ sudo apt-get install ffmpeg
Error: you are not in group sudoers
[email hpc-consult@list.arizona.edu]
[wait up to \infty days]

$ myutility
ffmpeg version 3.1.1 required, found ffmpeg
3.1.0
[loud swearing]

$ logout
```

What is a container?

- Isolates not only a process, but its dependencies and view of the filesystem
    - (Isn't that a VM? Well, almost. Containers don't involve hardware virtualization at all, and they all share a single kernel.)
- Defines an *immutable* filesystem with a set of packages / files installed
- Used for application packaging when dependencies are complicated

Artist's impression of containers on *Ocelote*

Container vocabulary

- Docker
  - A company and a piece of software and a container format
- Container image
  - Archive (like .zip or .tar.gz) containing a snapshot of the filesystem the contained programs will run in
- Dockerfile
  - Text-based recipe to build a container image from files and shell commands
- Singularity
  - Competing container format with less flexibility, but compatible with HPC permissions and quotas
- Container runtime
  - A piece of software that can load a container image and execute commands inside it (Docker, Singularity)

Building a container (the short version)

In your Dockerfile

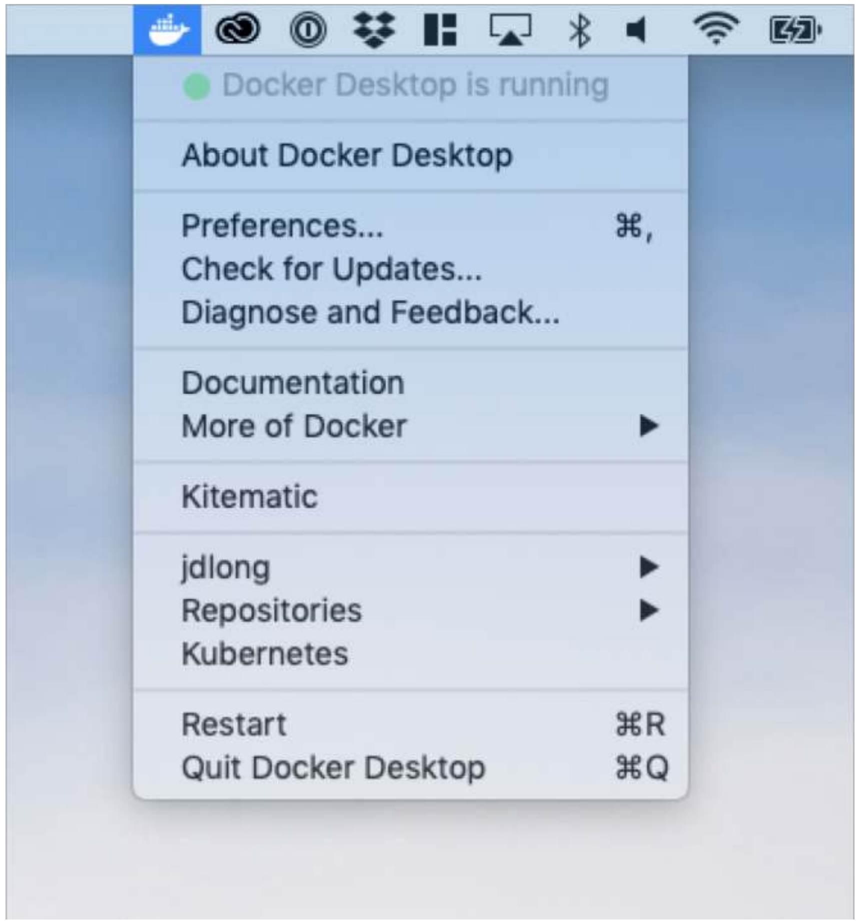
1. Choose a "base image" (`FROM`)
2. Add files (`ADD`)
3. Execute build commands (`RUN`)

In your shell:

1. `$ docker build ./ -t containername`
2. `$ docker run -it containername bash`

- Whether targeting Docker or Singularity, a Dockerfile is the way to go
  - https://docs.docker.com/engine/reference/builder/
- Substitute `bash` for any command, provided it's present within the container
- **UA HPC caveat:** use a CentOS 6.10 base image to ensure compatibility with the vintage OS on *El Gato* and *Ocelote*

Before we get started



- Ensure that Docker desktop is running
- If there's an option to log in on the menu, log in with your DockerHub (download) credentials
- Make sure the docker command works in your terminal

```
$ docker -v
Docker version 18.09.1,
build 4c52b90
```

Let’s build a container

- ffmepg is notoriously annoying to install, and isn’t even in the CentOS package repository
    - Pretend we can’t just `module load ffmpeg`…
- We need ffmpeg to generate animations from matplotlib
- What if we could package ffmpeg *and* Python *and* our script to run on HPC?
- But first, let’s just make the simplest possible (empty) container…

1. Make a new directory (e.g. container-example/)
2. Make a new file named `Dockerfile` with the contents
   ```
   FROM centos:6.10
   ```
3. Build it: `docker build . -t example`
4. Run bash in your container:
   ```
   docker run -it example bash
   ```
5. Now you’re running in CentOS 6.10*!
   Try `cat /etc/redhat-release`

*Well, kinda.

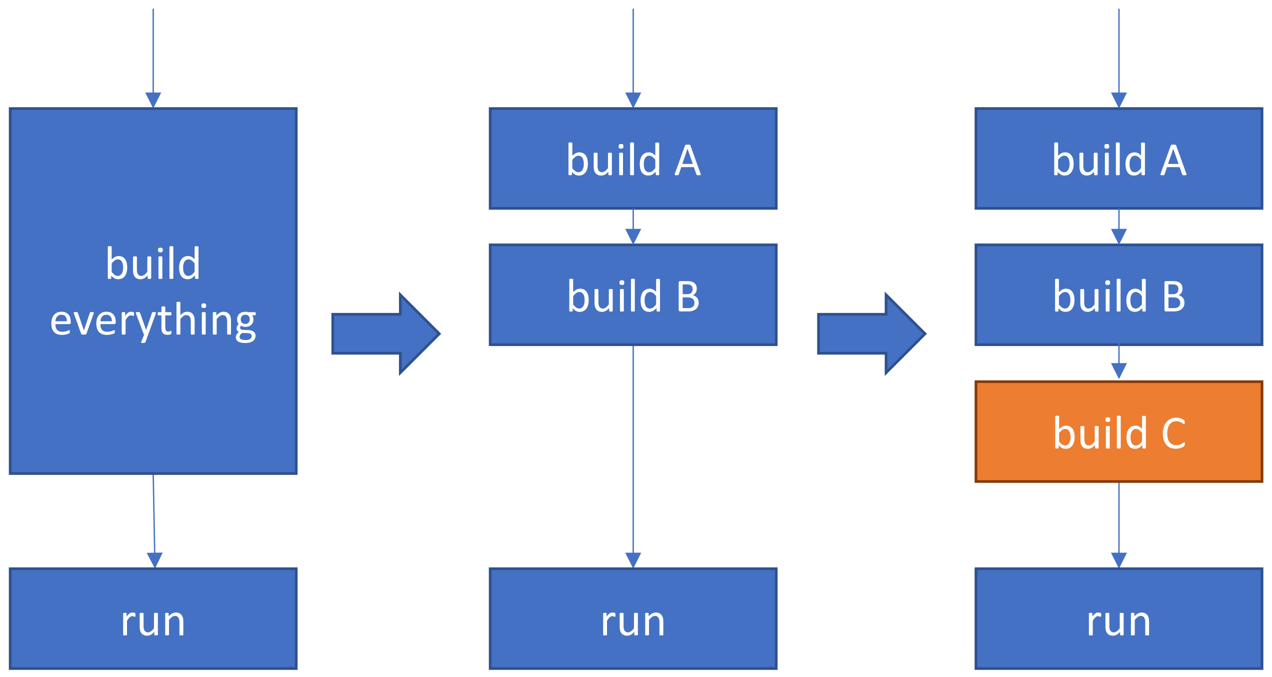
Why CentOS 6.10?

- What is CentOS anyway?
  - “Community Enterprise OS”, basically a $0 version of Red Hat Enterprise Linux
  - Slow moving, conservative, loved by systems administrators
  - Loathed by scientists
- Uses really old versions of everything and tries to keep things totally backwards compatible
- If you run an old, compiled program on a newer Linux, it’s practically guaranteed to work.
- If you run a new program on old Linux, you may see “FATAL: kernel too old”
- Short answer:

To make UA HPC happy

Adding layers

- Layers are a clever way to separate the process of building a container into stages
- Each Dockerfile line adds a layer
- Adding new steps to the **end** of the Dockerfile reuses previous layers



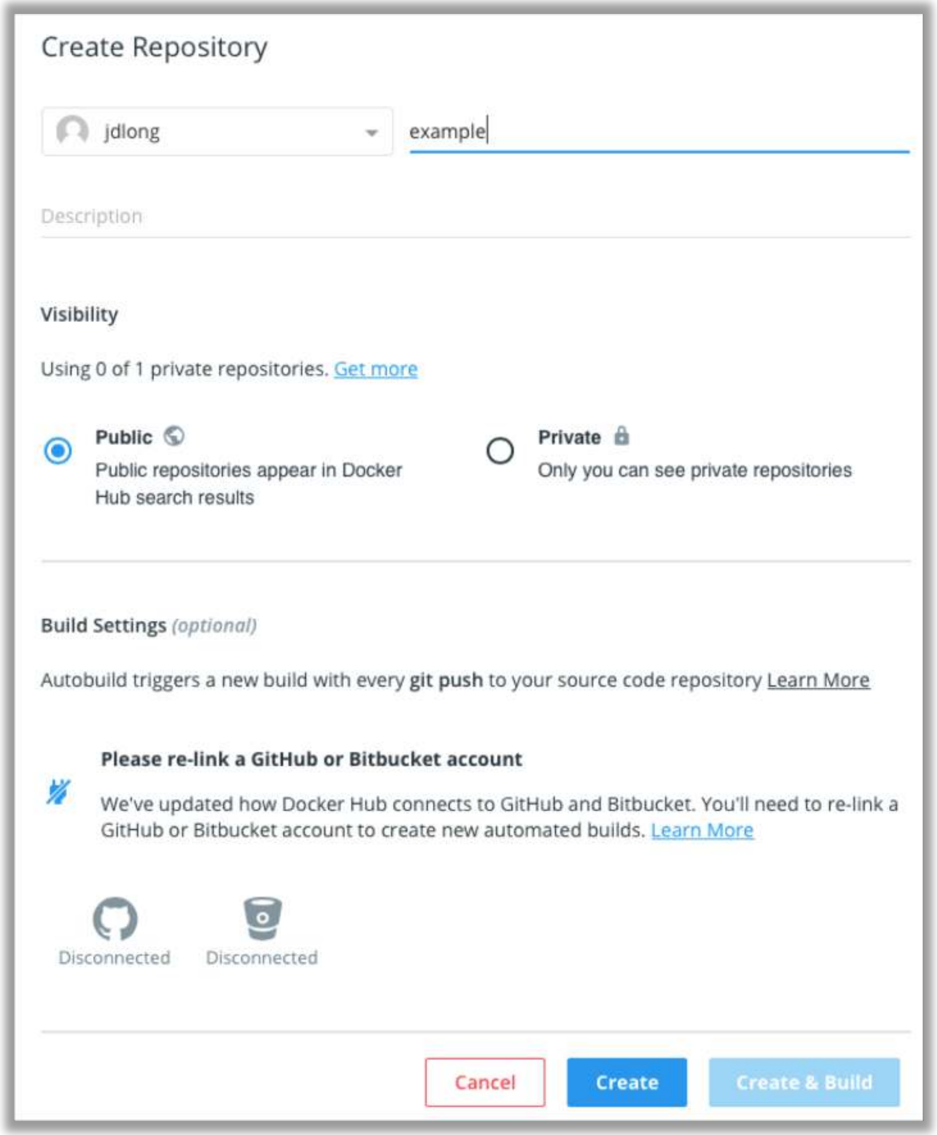
Adding layers

- Following https://www.vultr.com/docs/how-to-install-ffmpeg-on-centos, we add RUN directives to our Dockerfile
- Everything runs as root by default, so we can omit sudo
    - Does this worry you? It should…
- We can also ignore the line about shutting down and rebooting
- Build it: `docker build . -t example`
- See if ffmpeg runs: `docker run -it example ffmpeg`

```
FROM centos:6.10

RUN yum install epel-release -y

RUN yum update -y

RUN rpm --import
http://li.nux.ro/download/nux/RPM-GPG-KEY-
nux.ro

RUN rpm -Uvh
http://li.nux.ro/download/nux/dextop/el6/x86_6
4/nux-dextop-release-0-2.el6.nux.noarch.rpm

RUN yum install ffmpeg ffmpeg-devel -y
```

Moving from your laptop to UA HPC

- The following steps require an HPC account and a Dockerhub account
- Open https://hub.docker.com/
- Make a new repository called “example”
  - The full name will be something like “jdlong/example” where “jdlong” is your DockerHub username
- Locally, open a terminal and type
  `docker login`
- Now we want to build and tag with the new, full name:
  `docker build . -t jdlong/example`
- And finally, push:
  `docker push jdlong/example`


Create Repository

jdlong / example

Description

Visibility

Using 0 of 1 private repositories. Get more

[✓ Public] Public repositories appear in Docker Hub search results

Private: Only you can see private repositories

Build Settings *(optional)*

Autobuild triggers a new build with every **git push** to your source code repository Learn More

Please re-link a GitHub or Bitbucket account

We've updated how Docker Hub connects to GitHub and Bitbucket. You'll need to re-link a GitHub or Bitbucket account to create new automated builds. Learn More

Moving from your laptop to UA HPC

- Log in to your preferred HPC cluster
- Enable the Singularity container runtime
  - module load singularity
- Download your container
  - singularity pull docker://jdlong/example
- The Singularity image is now present at ./example_latest.sif
  - To run a command in the container, ./example_latest.sif ffmpeg

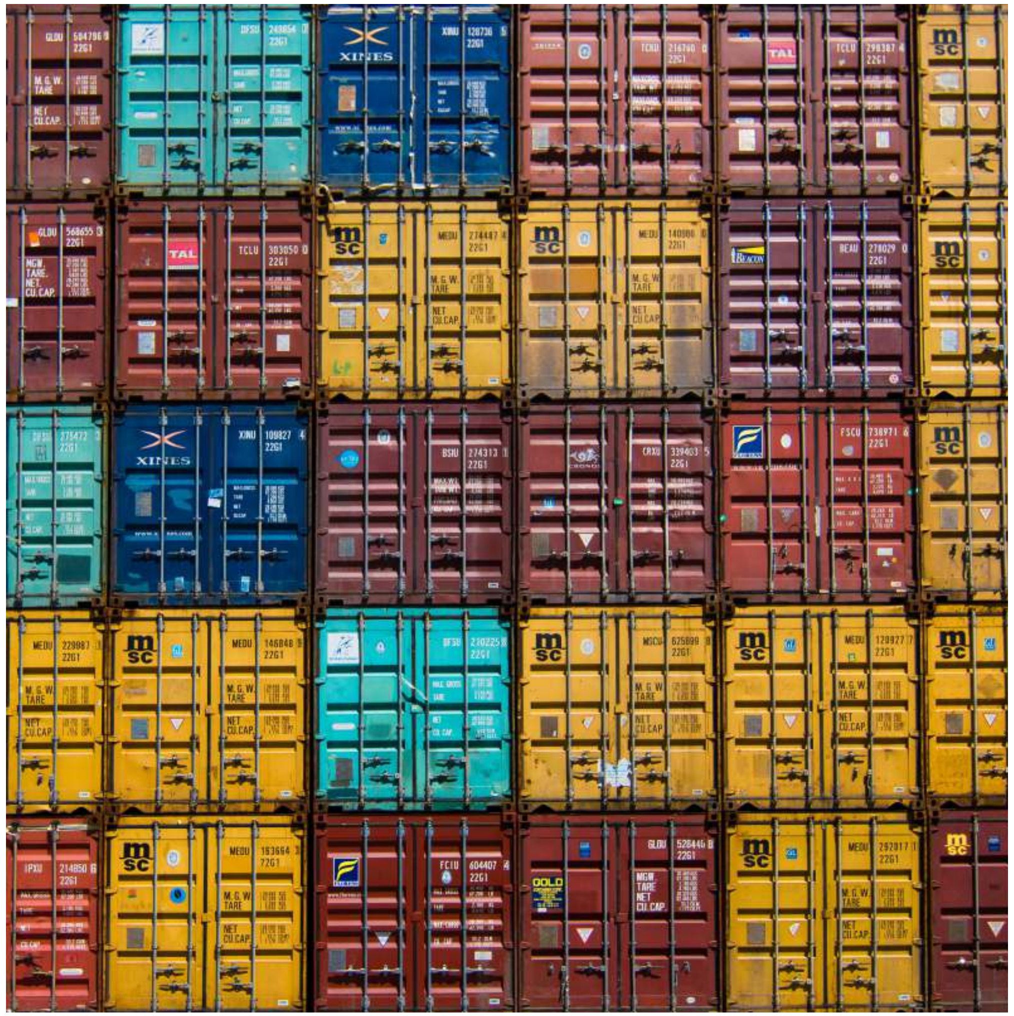


Photo by Guillaume Bolduc on Unsplash

Why is all this useful?

- These steps work on any machine with a container runtime, no matter the underlying OS
  - (As long as it’s not older than Linux 2.6…)
- You can distribute instructions to automatically and exactly recreate your computing environment along with your papers / software
- Defining a set of packages for a teaching environment or demo
  - e.g. https://mybinder.org
- Running software that makes assumptions about its environment that conflict with other software you need
  - e.g. some software needs version 1.0 of a library, other software uses 2.0