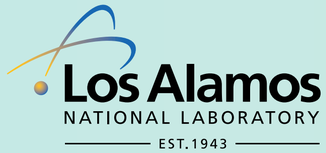




SC23
Denver, CO | i am hpc.

# Charliecloud's layer-free, Git-based container build cache

**Reid Priedhorsky (reidpr@lanl.gov)**,[1] Jordan Ogas,[1] Claude H. (Rusty) Davis IV,[1]
Z. Noah Hounshel,[1,2] Ashlyn Lee,[1,3] Benjamin Stormer,[1,4] R. Shane Goff[1]

[1]Los Alamos National Laboratory [2]U. North Carolina Wilmington [3]Colorado State U. [4]U. Texas at Austin

LA-UR 23-32773

# Charliecloud

## LANL's lightweight, fully unprivileged container impl.

- ¼ the size of next smallest competitor w/ better HPC fit

## Project goals:

- Deliver containers for production HPC applications
- Study future container needs
- Influence the industry so it's closer to what we want

## "Philosophy of Charliecloud":

- Be transparent, not opaque
- Be simple, not complex
- Be unprivileged

# Charliecloud team (this work and current)

**Reid Priedhorsky**
HPC-ENV

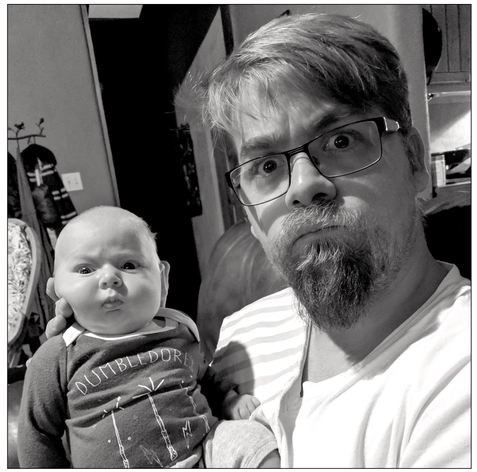
**Jordan Ogas**
XCP-1

**Rusty Davis**
HPC-DES

**Z. Noah Hounshel**
HPC-ENV / UNC Wilmington

**Ashlyn Lee**
HPC-ENV / CO State U.

**Ben Stormer**
HPC-ENV / UT Austin



**Shane Goff**
HPC-DES

**Lucas Caudill**
HPC-ENV

**Megan Phinney**
HPC-ENV

**Layton McCafferty**
HPC-ENV / MT State

# No layers

Charliecloud focuses on *image states*, created one after the other as Dockerfile is interpreted

- this is a “version-oriented” view

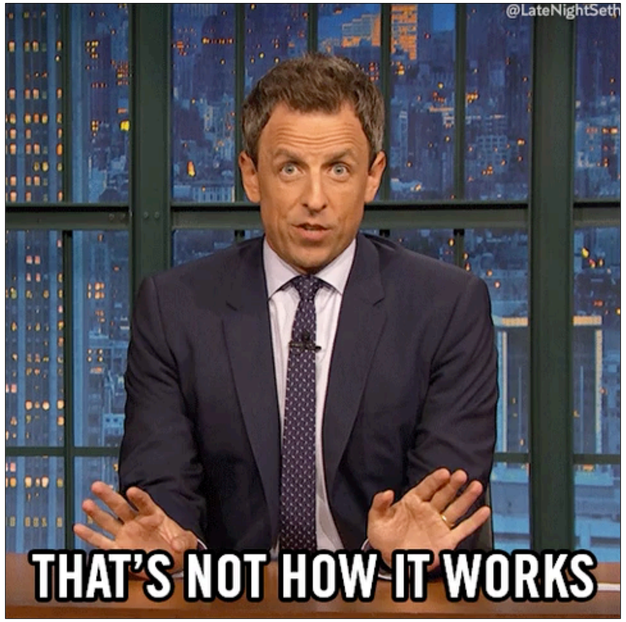
Alternately and more common:
focus on the *transitions* between states

- i.e. *diffs* or *patches* or *layers*
- “change-oriented” view
- OCI tarballs are just a diff format
  new/changed files in full
  whiteouts for removed files

**These views are equivalent.**

- they have the same descriptive power
- you can compute one from the other

# Build cache assumptions (any implementation)




1. Retrieving state is cheaper than executing
2. State depends only on *visible input* and *ancestry*
   - ☞ i.e., previous state plus the new instruction

"`RUN true`" ⇒ probably faster to execute
"`RUN date > foo`" ⇒ depends on the time too

1. Ignore the problem
   - ☞ i.e., assumption is close enough
2. Retrieve minimum state needed
   - ☞ i.e., only retrieve state of *last* cache hit
3. Make the build cache optional
   - ☞ `--no-cache`, `--rebuild`

Charliecloud

mitigations

# Wacky new idea: Use Git for container build cache

Image states are a tree ⇒ smells like version control

Git seemed like a good choice

- ☞ very widely used (incl. by us), lots of expertise out there
- ☞ performant, scalable, well tested

Disadvantages we worked around, e.g.

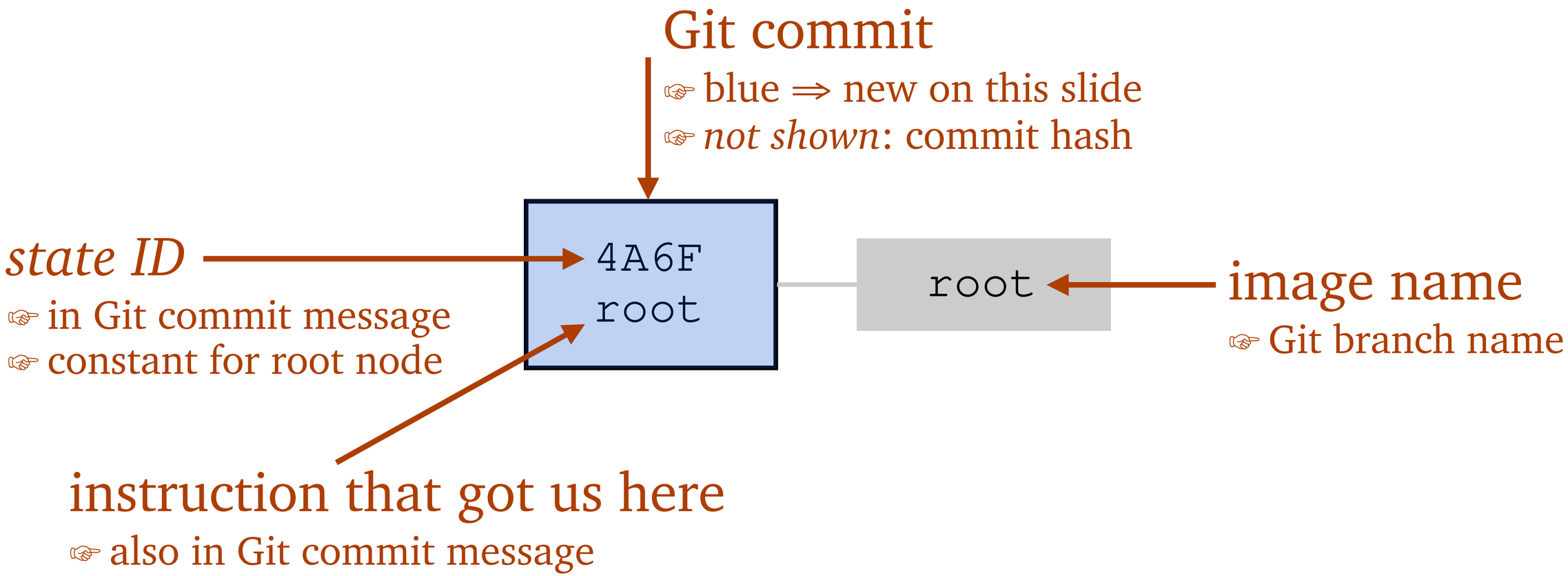
- ☞ metadata, empty directories ignored, problem file types
- ☞ filenames starting with `.git` are special

# *II. How the build cache works*

(by example)

# Empty cache

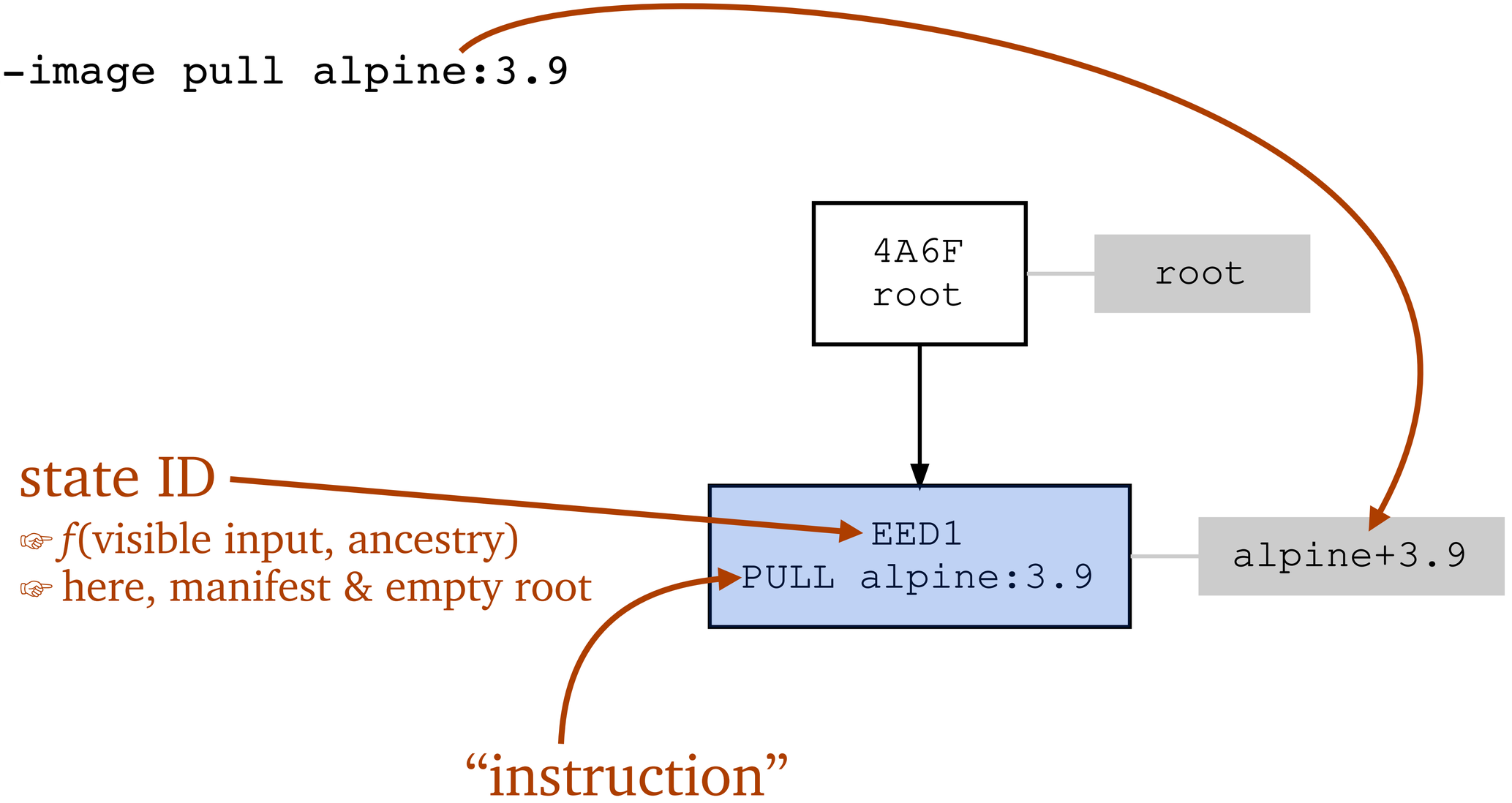
```
$ ch-image build-cache --reset
```




“`ch-image build-cache --dot`” yields PDF above
(except for blue highlights)

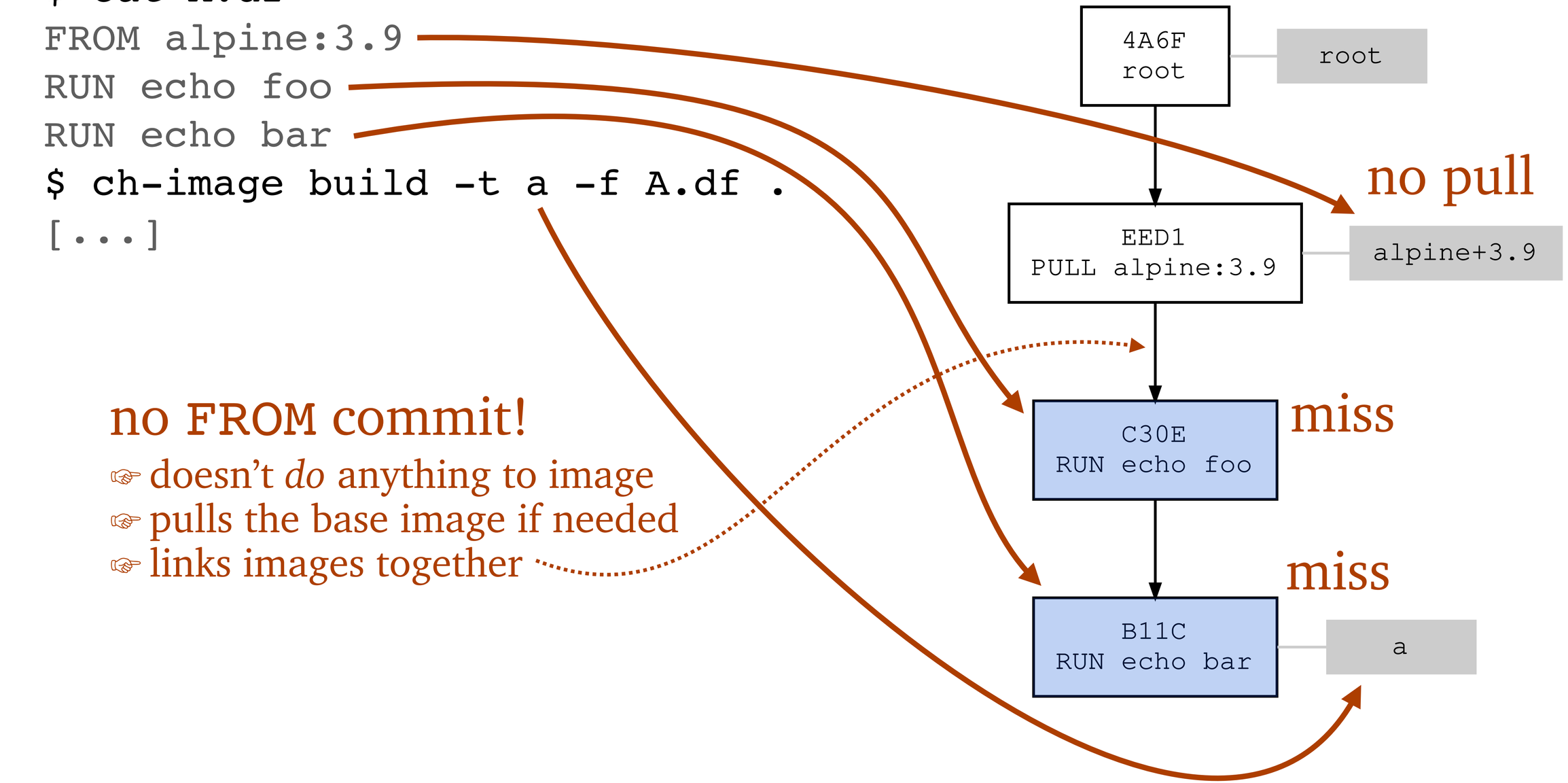
# Initial pull

```
$ ch-image pull alpine:3.9
```



# Build a Dockerfile

```
$ cat A.df
FROM alpine:3.9
RUN echo foo
RUN echo bar
$ ch-image build -t a -f A.df .
[...]
```

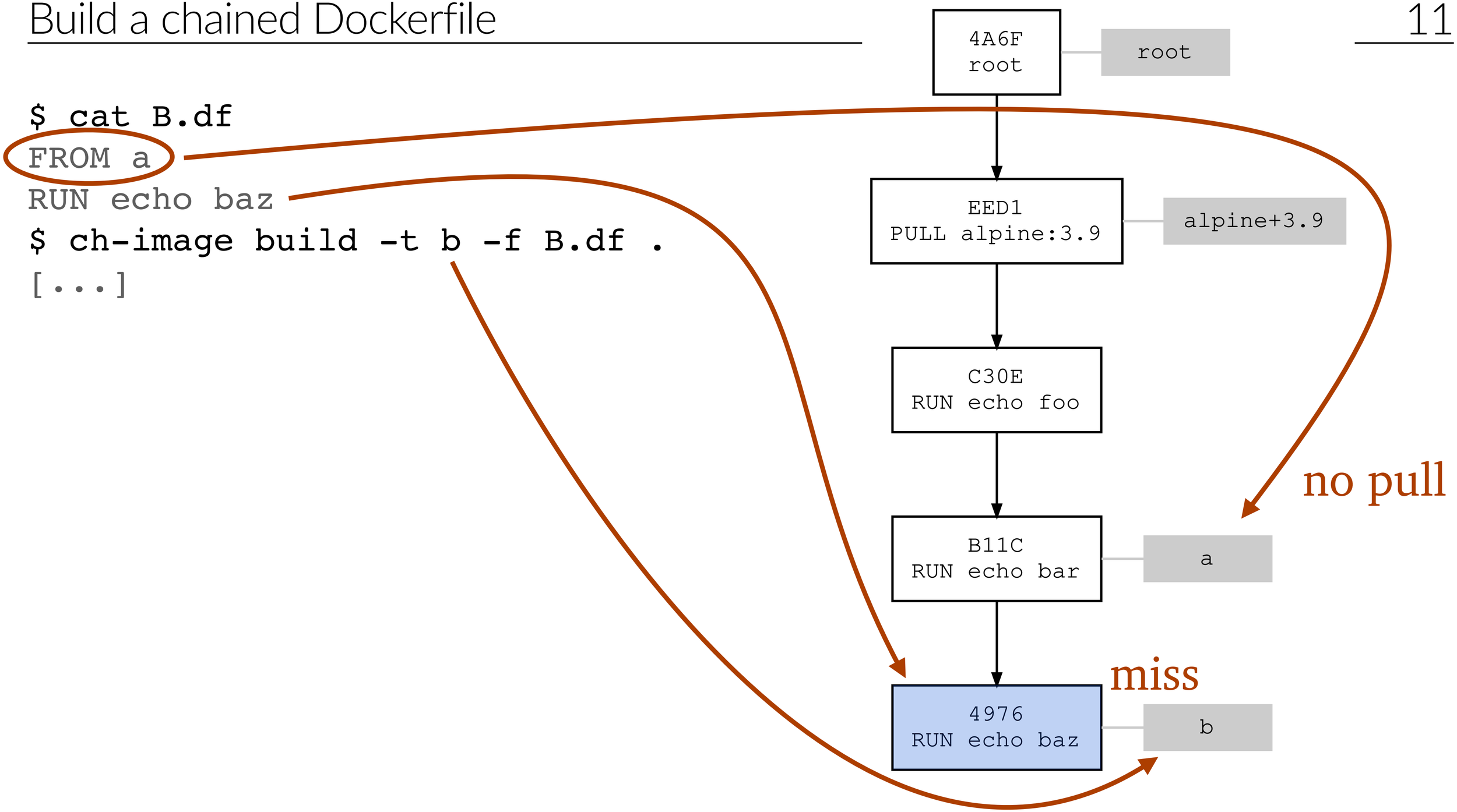



## no FROM commit!

- doesn't *do* anything to image
- pulls the base image if needed
- links images together

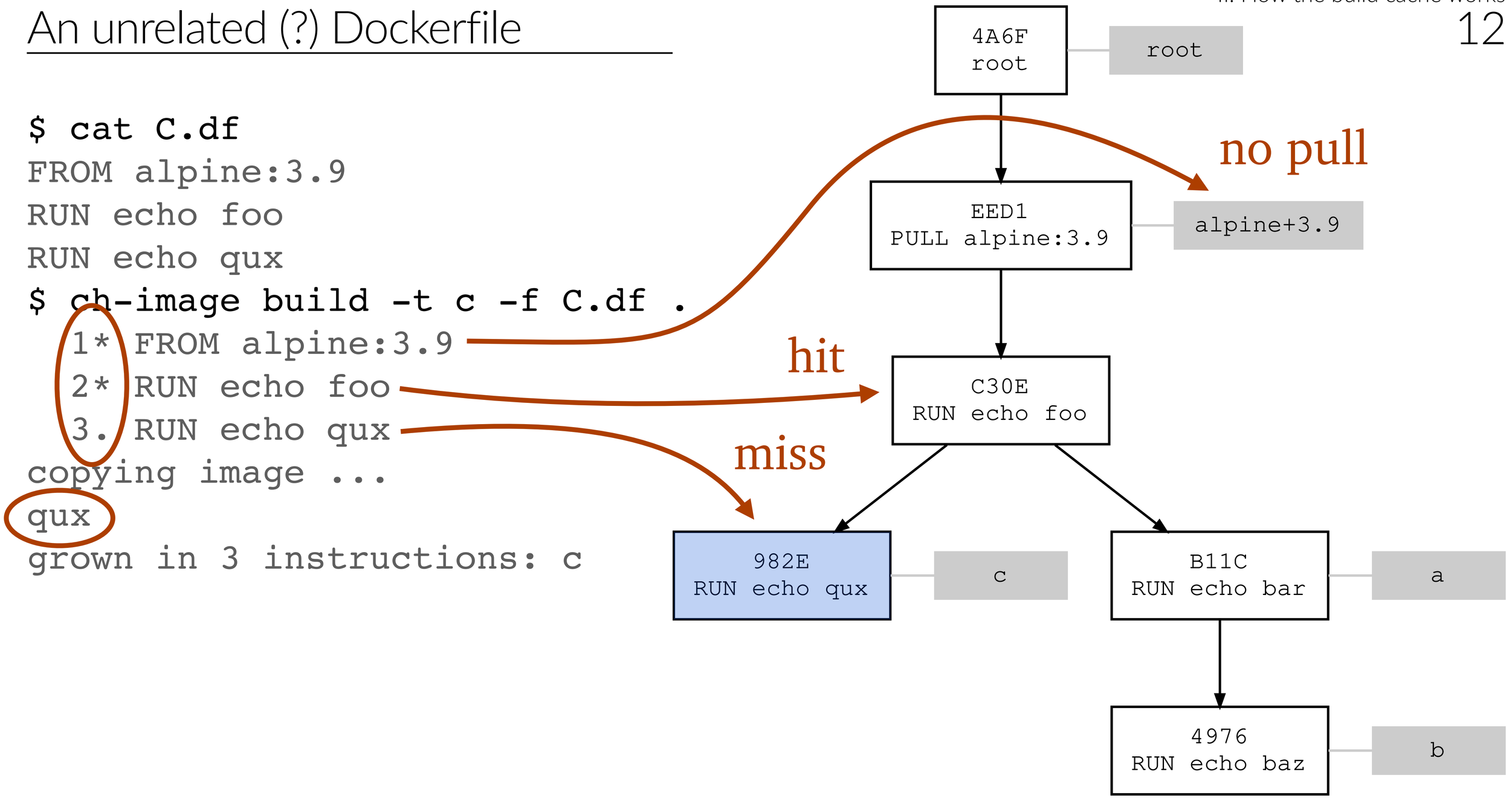
# Build a chained Dockerfile

```
$ cat B.df
FROM a
RUN echo baz
$ ch-image build -t b -f B.df .
[...]
```



# An unrelated (?) Dockerfile

```
$ cat C.df
FROM alpine:3.9
RUN echo foo
RUN echo qux
$ ch-image build -t c -f C.df .
  1* FROM alpine:3.9
  2* RUN echo foo
  3. RUN echo qux
copying image ...
qux
grown in 3 instructions: c
```



# *III. Performance*

# *Late-breaking development!!*

Build cache can store large files out-of-band (not Git).

- ☞ uses hard links — a fast metadata-only operation

We assumed hard links are copy-on-write (COW).

Hard links are not copy-on-write. 🤦

- ☞ i.e., use of large files easily corrupts your cache
- ☞ default is no large files (whew!)


Results use large-file threshold of 4MiB.

However, <u>results should hold</u> given right FS

- ☞ no evidence of corruption in experiments so far
- ☞ large files now use “reflinks” (copy metadata, share data)
  requires BTRFS, XFS, ZFS or maybe a few others
- ☞ metadata-only operation which *is* COW

# Median build time on ext4 (cold, warm)

Charliecloud source code or synthetic for this experiment


| *shading* |
|---|
| 4× faster, –75% |
| equal |
| 4× slower, +300% |
| *than Charliecloud* |

| temp | image | Charliecloud time | docker time | docker vs. ch | podman time | podman vs. ch | Charliecloud is … |
|---|---|---|---|---|---|---|---|
| cold | megafiles | 76 s | 45 s | −41 % | 97 s | +28 % | slower |
| | mini | 9.4 s | 7.8 s | −17 % | 6.4 s | −31 % | |
| | almalinux | 54 s | 52 s | −4.4 % | 55 s | +1.1 % | |
| | megapkg | 490 s | 480 s | −2.4 % | 550 s | +11 % | |
| | openmpi | 410 s | 410 s | +0.048 % | 420 s | +1.4 % | about the same |
| | paraview | 930 s | 940 s | +0.73 % | 960 s | +3.6 % | |
| | micro | 0.67 s | 0.83 s | +23 % | 0.77 s | +14 % | |
| | megabytes | 21 s | 31 s | +46 % | 70 s | +230 % | faster |
| | megainst | 9.4 s | 46 s | +390 % | 68 s | +630 % | much faster |
| warm | megafiles | 48 s | 28 s | −42 % | 46 s | −3.3 % | slower |
| | mini | 1.2 s | 0.51 s | −57 % | 0.61 s | −49 % | |
| | almalinux | 4.8 s | 2.1 s | −56 % | 2.4 s | −51 % | |
| | megapkg | 300 s | 300 s | −0.71 % | 340 s | +14 % | |
| | openmpi | 250 s | 250 s | +1.8 % | 260 s | +2.6 % | about the same |
| | paraview | 440 s | 450 s | +3.6 % | 460 s | +4.7 % | |
| | micro | 0.39 s | 0.52 s | +36 % | 0.59 s | +53 % | |
| | megabytes | 9.3 s | 17 s | +88 % | 33 s | +260 % | faster |
| | megainst | 5.0 s | 24 s | +370 % | 40 s | +710 % | much faster |

# Median build time on ext4 (hot)


| *shading* |
|---|
| 4× faster, −75% |
| equal |
| 4× slower, +300% |
| *than Charliecloud* |

| temp | image | Charliecloud time | docker time | docker vs. ch | podman time | podman vs. ch |
|---|---|---|---|---|---|---|
| hot | megafiles | 13 s | 0.21 s | −98 % | 0.35 s | −97 % |
| | mini | 0.81 s | 0.21 s | −74 % | 0.16 s | −81 % |
| | almalinux | 1.7 s | 0.21 s | −87 % | 0.19 s | −89 % |
| | megapkg | 13 s | 0.22 s | −98 % | 0.28 s | −98 % |
| | openmpi | 2.0 s | 0.22 s | −89 % | 0.71 s | −65 % |
| | paraview | 3.2 s | 0.22 s | −93 % | 0.91 s | −72 % |
| | micro | 0.32 s | 0.22 s | −31 % | 0.16 s | −50 % |
| | megabytes | 0.45 s | 0.21 s | −52 % | 0.35 s | −22 % |
| | megainst | 0.68 s | 0.24 s | −64 % | 5.8 s | +750 % |

Charliecloud is …

a lot slower

☞ baseline is *much* faster

???

7 minutes cold

# Disk usage totals

## Charliecloud's storage is roughly ⅓× to 2½× the size of Docker

| | | | ch | | | | | ch compacted | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | docker | everything | | w/o unpacked | | | everything | | w/o unpacked | |
| **temp** | **image** | MiB | MiB | vs. dko | MiB | vs. dko | time | MiB | vs. dko | MiB | vs. dko |
| | micro | 6.5 | 20 | +210 % | 8.7 | +34 % | 0.35 s | 17 | +160 % | 5.7 | −13 % |
| | mini | 190 | 570 | +200 % | 250 | +34 % | 4.0 s | 480 | +160 % | 160 | −13 % |
| | megainst | 12 | 26 | +120 % | 15 | +22 % | 1.0 s | 17 | +43 % | 5.8 | −52 % |
| | megafiles | 4200 | 8400 | +100 % | 4200 | — | 82 s | 8300 | +100 % | 4200 | — |
| **cold** | almalinux | 590 | 1200 | +100 % | 620 | +4.3 % | 10 s | 940 | +58 % | 360 | −39 % |
| | openmpi | 930 | 1600 | +71 % | 940 | +2.1 % | 18 s | 1200 | +27 % | 530 | −42 % |
| | paraview | 2100 | 3300 | +54 % | 2100 | — | 33 s | 2500 | +16 % | 1300 | −37 % |
| | megapkg | 7800 | 9600 | +23 % | 4700 | −40 % | 18 s | 9600 | +23 % | 4700 | −40 % |
| | megabytes | 4200 | 4200 | +1.4 % | 4200 | — | 1.7 s | 4200 | — | 4100 | — |
| | micro | 6.6 | 17 | +160 % | 5.7 | −14 % | 0.22 s | 17 | +160 % | 5.7 | −15 % |
| | mini | 190 | 480 | +160 % | 160 | −13 % | 0.82 s | 480 | +160 % | 160 | −13 % |
| | megainst | 14 | 21 | +48 % | 9.1 | −35 % | 1.2 s | 17 | +24 % | 5.8 | −58 % |
| | megafiles | 6200 | 10 000 | +68 % | 6300 | — | 100 s | 10 000 | +68 % | 6200 | — |
| **warm** | almalinux | 590 | 940 | +58 % | 370 | −38 % | 2.9 s | 940 | +58 % | 360 | −39 % |
| | openmpi | 1100 | 1200 | +12 % | 550 | −48 % | 9.7 s | 1200 | +11 % | 540 | −49 % |
| | paraview | 2600 | 2700 | +4.8 % | 1500 | −40 % | 16 s | 2600 | +2.4 % | 1500 | −42 % |
| | megapkg | 13 000 | 9800 | −23 % | 4900 | −62 % | 28 s | 9800 | −24 % | 4800 | −62 % |
| | megabytes | 6200 | 6200 | — | 6200 | — | 0.51 s | 6200 | — | 6200 | — |




*shading*

4× smaller, –75%

equal

4× bigger, +300%

*than Docker*

# *IV. Implications*

# Git cache vs. overlay cache

| | factor | docker podman overlay summary | Charliecloud Git summary | winner |
|---|---|---|---|---|
| empirical | build time | ✅ usually fine<br>❌ some gotchas | ✅ usually fine<br>❌ some gotchas | *varies* |
| empirical | disk usage | ✅ usually fine?<br>❌ easy to store redundant layers | ✅ usually fine<br>❌ some obvious inefficiencies<br>❌ must garbage-collect for full benefit | *varies* |
| structural | diff format | ❌ tar poorly standardized<br>❌ tar not designed for diffs<br>❌ change one byte? ⇒ store the whole file again<br>❌ delete a file? ⇒ can't represent, need whiteouts | ✅ purpose-built versioned filesystem | Git |
| structural | cache overhead | ❌ increases with every instruction<br>❌ imposed on every file metadata operation | ✅ ~independent of instruction count<br>✅ instruction commit only<br>✅ roughly proportional to commit size | Git |
| structural | de-duplication | ❌ between ancestor/descendant images<br>❌ for same-named files | ✅ across all images in cache<br>✅ identical files at commit time<br>❌ similar files at garbage collect time<br>❌ large files have ancestry constraint | Git |

wacky new idea is competitive!

*no more minimizing instruction count!?*

wacky new idea wins?

Charliecloud

Los Alamos NATIONAL LABORATORY EST. 1943

SC23 Denver, CO | i am hpc.

# Charliecloud's layer-free, Git-based container build cache


**Reid Priedhorsky (reidpr@lanl.gov)**,[1] Jordan Ogas,[1] Claude H. (Rusty) Davis IV,[1]
Z. Noah Hounshel,[1,2] Ashlyn Lee,[1,3] Benjamin Stormer,[1,4] R. Shane Goff[1]

[1]Los Alamos National Laboratory [2]U. North Carolina Wilmington [3]Colorado State U. [4]U. Texas at Austin



LA-UR 23-32773