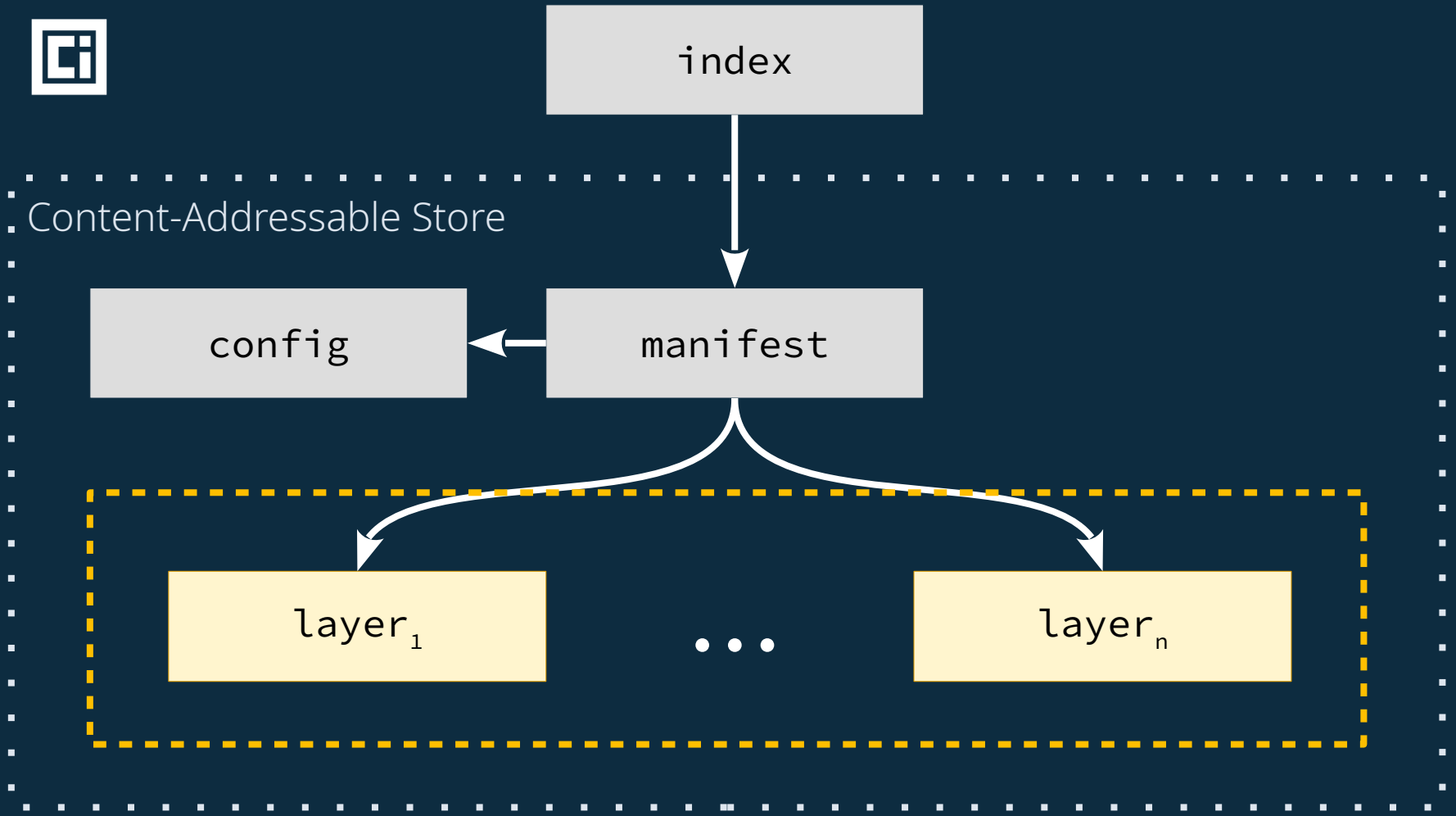
Container Images Considered Harmful

(... and some things we can do about it.)

Aleksa Sarai
Senior Software Engineer
@lordcyphar
<cyphar@cyphar.com>

An Ideal Image Format

Deduplicated as much as possible (*transfer* and *storage*)
Parallelisable (*transfer* and *storage*)
Reproducible (and have a canonical representation)
Non-avalanching
Transparent


index
Content-Addressable Store
config
manifest
layer1
...
layern

What's Wrong With Tar?

- A fair bit.

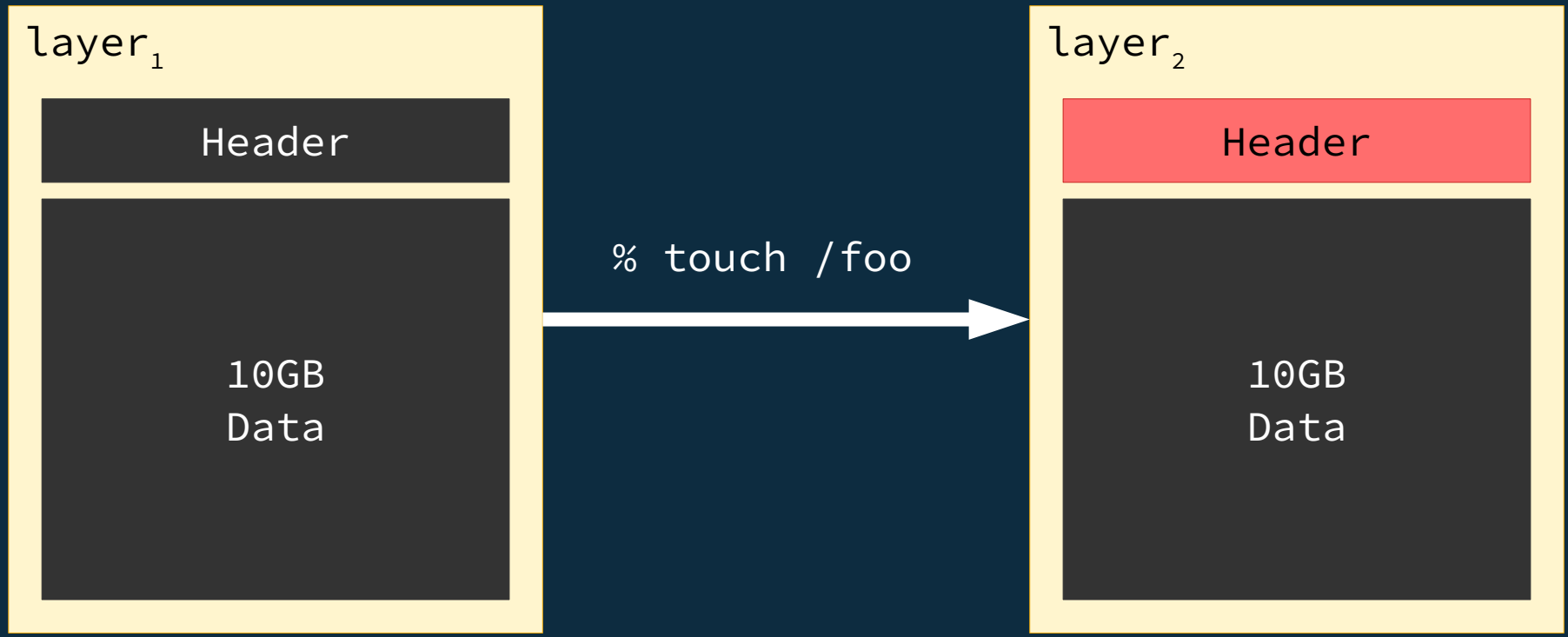
An Ideal Image Format

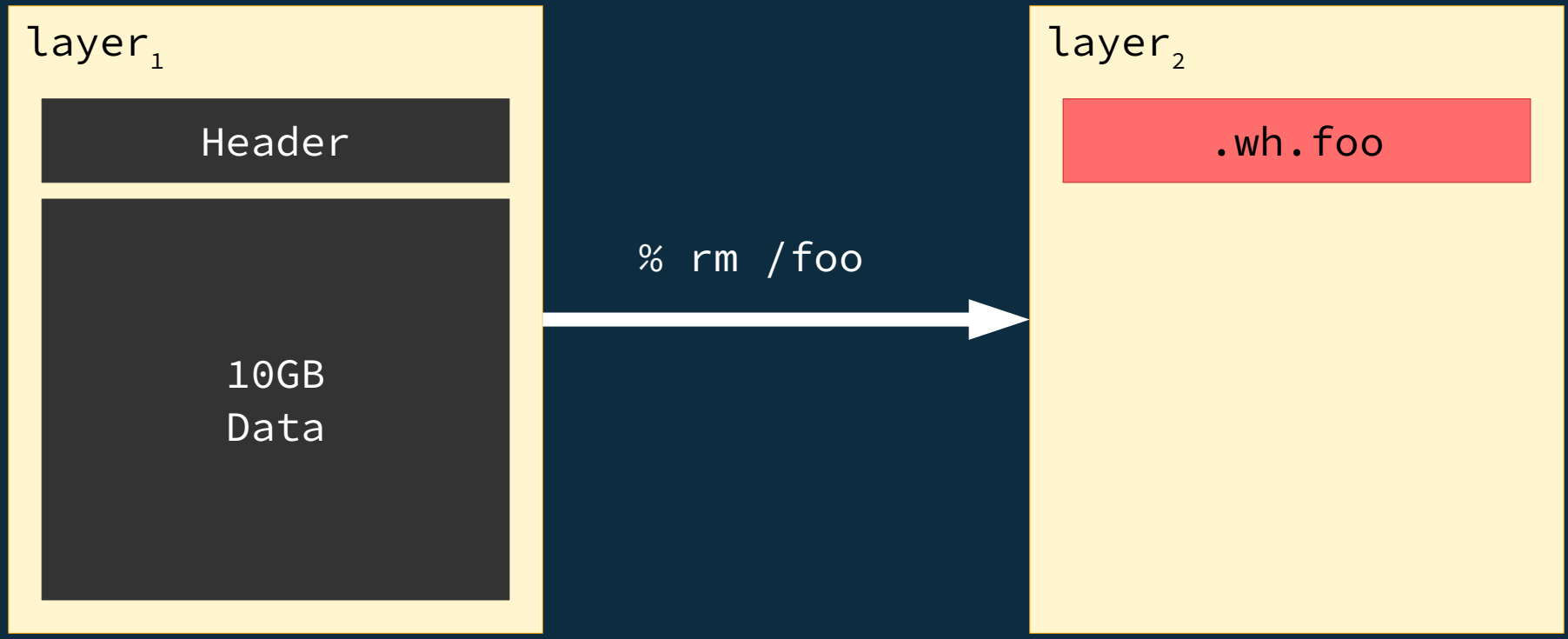
Deduplicated as much as possible (*transfer* and *storage*)
Parallelisable (*transfer* and *storage*)
Reproducible (have a canonical representation)
“Non-avalanching”
Transparent

What Tar Gives Us

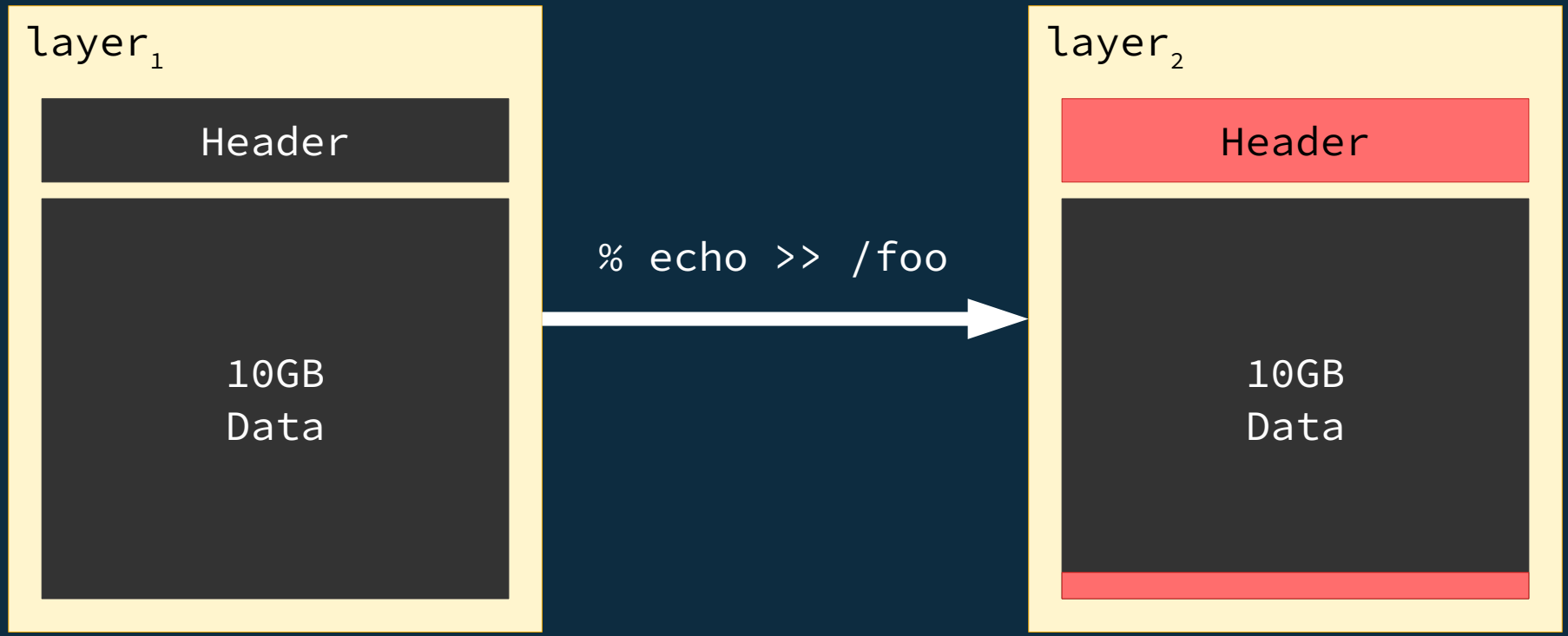
What Tar Gives Us

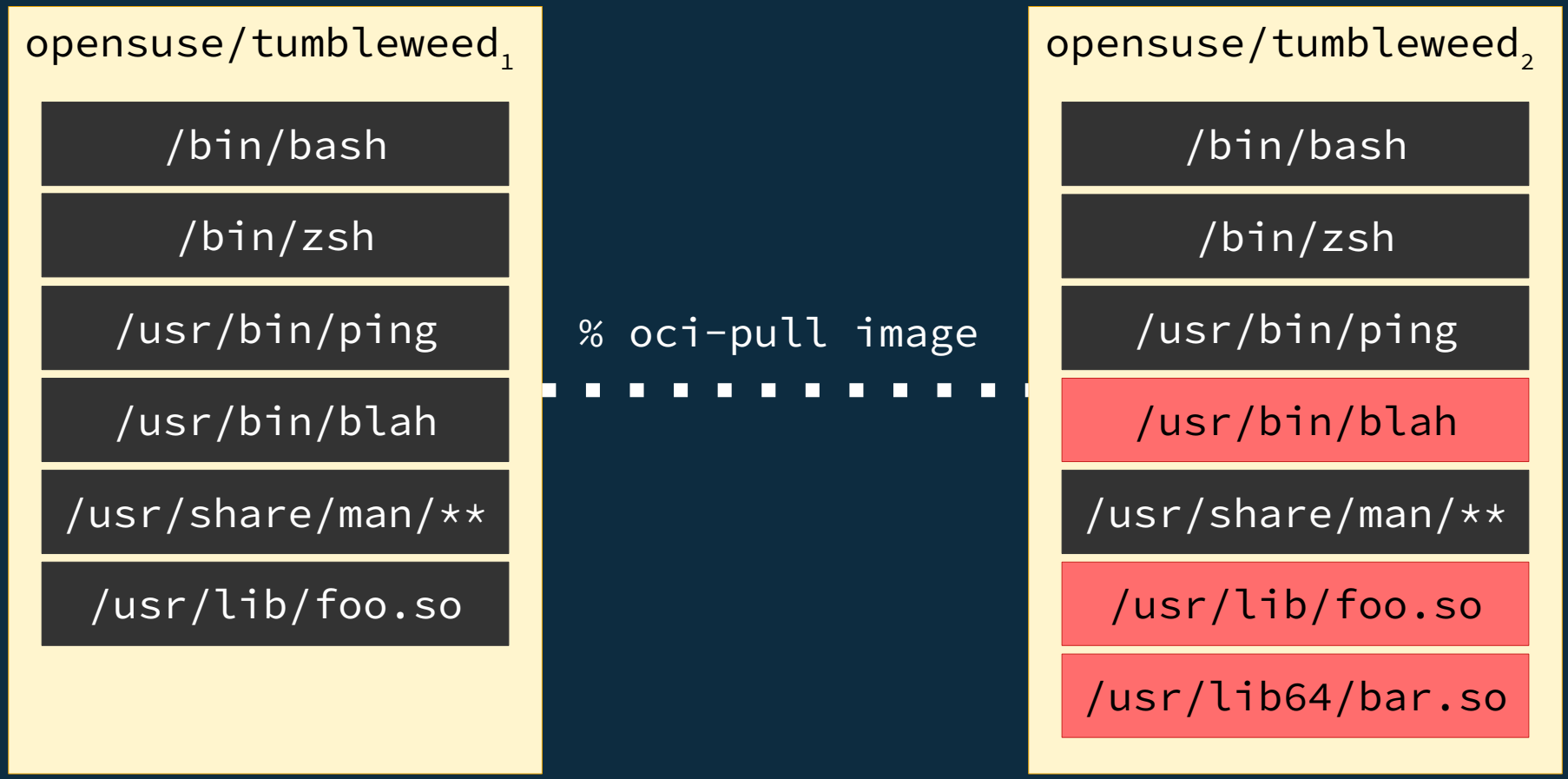
(This page intentionally left blank.)

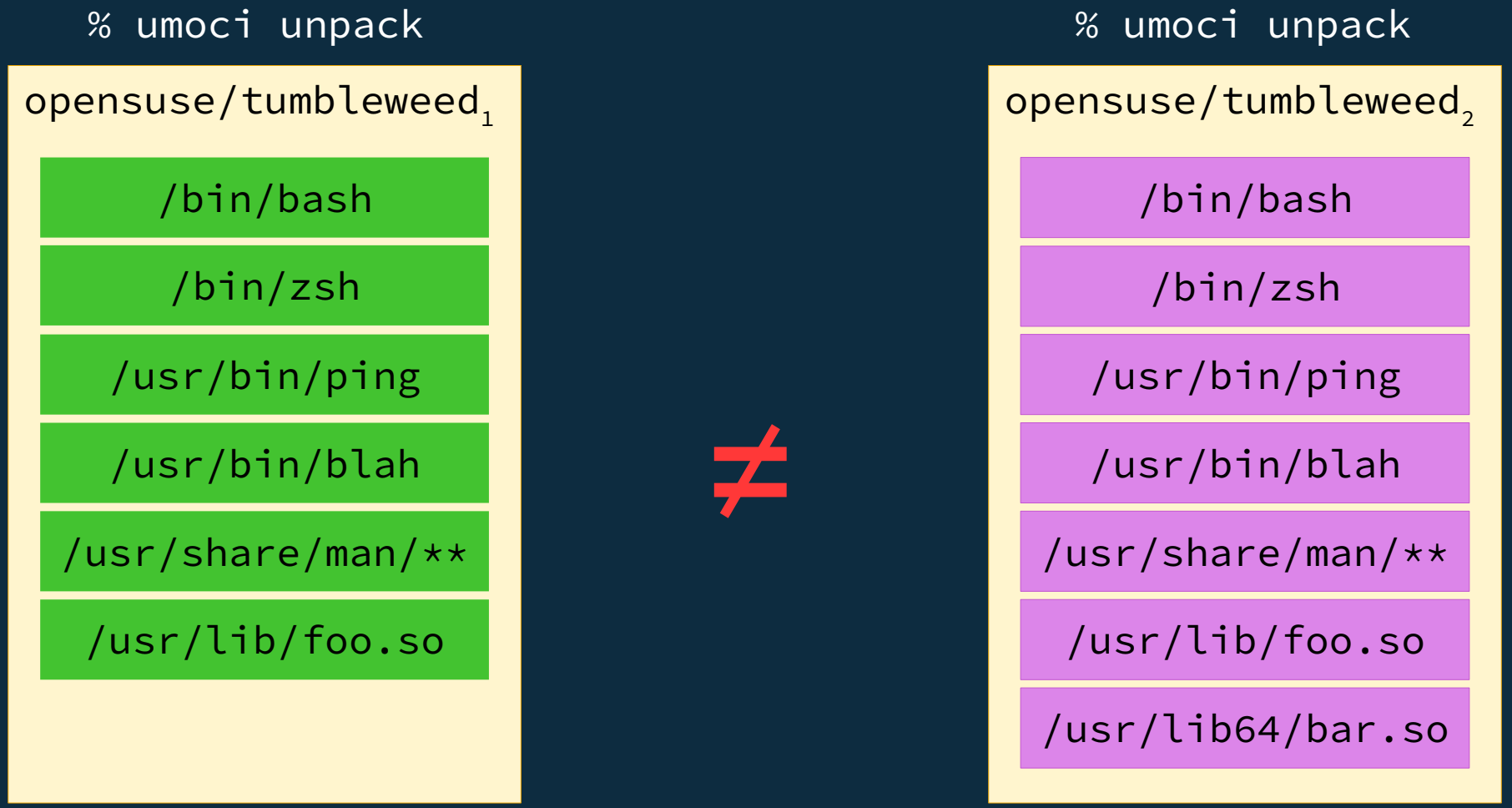

layer1
Header
10GB
Data
% touch /foo
layer2
Header
10GB
Data

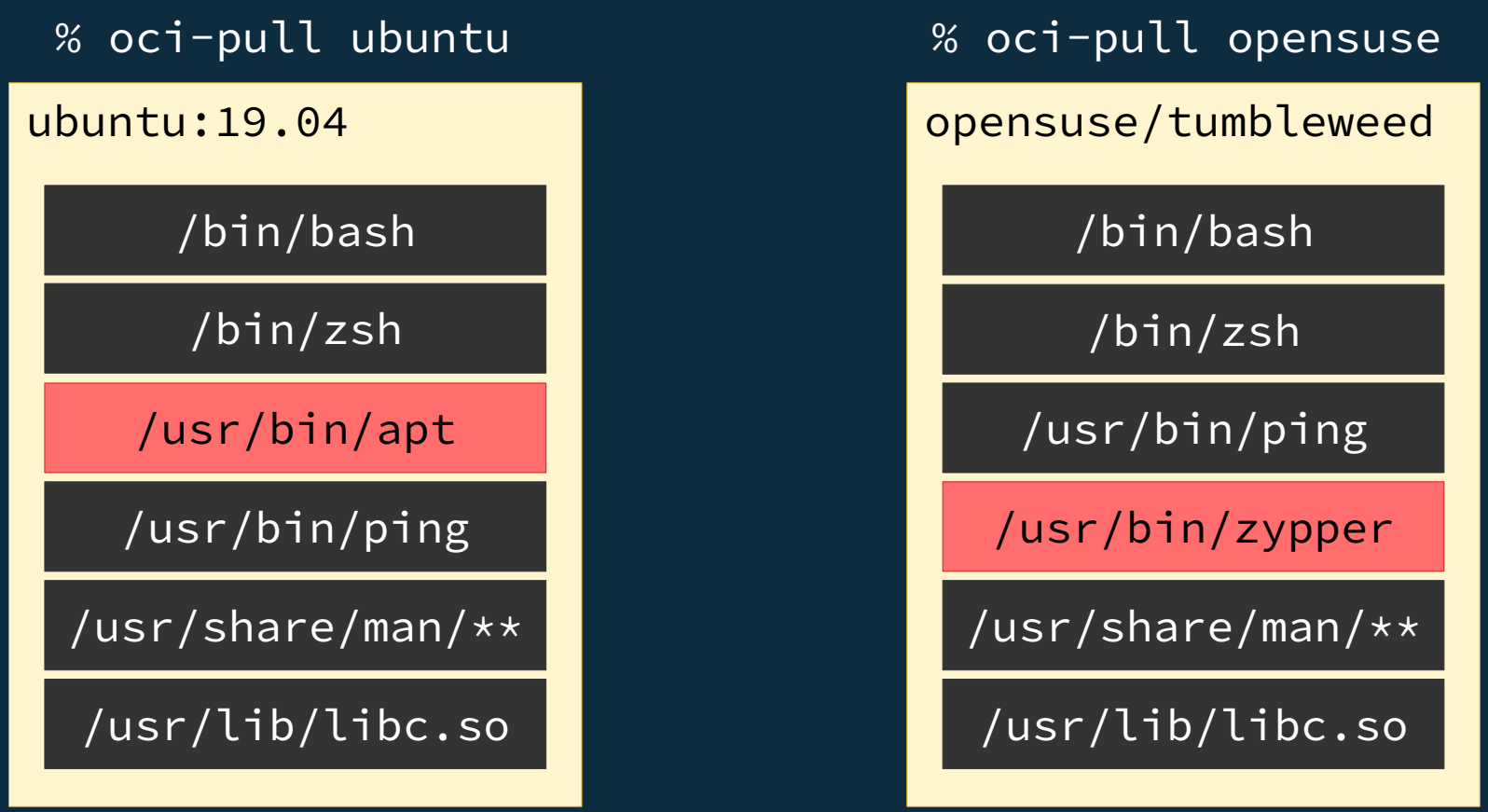


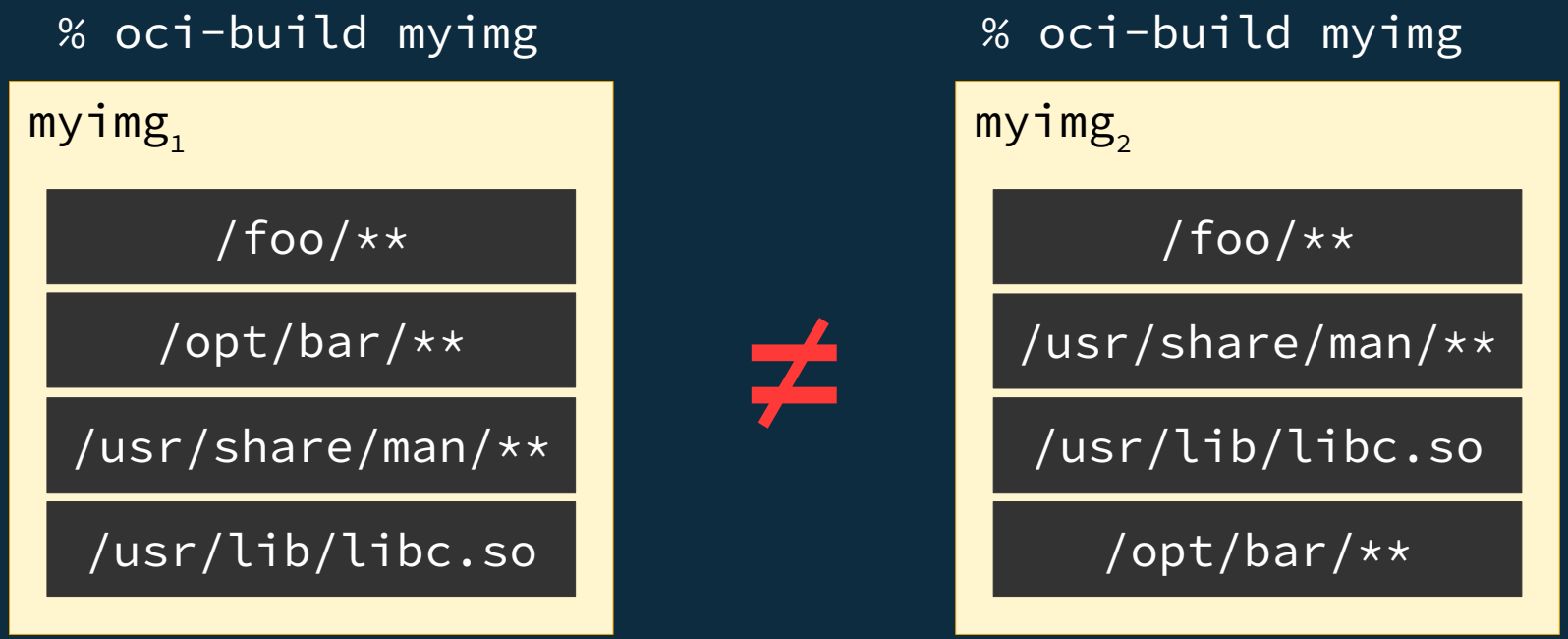
I'm not pranking you – your image gets **bigger**.


layer1
Header
10GB
Data
% echo >> /foo
layer2
Header
10GB
Data

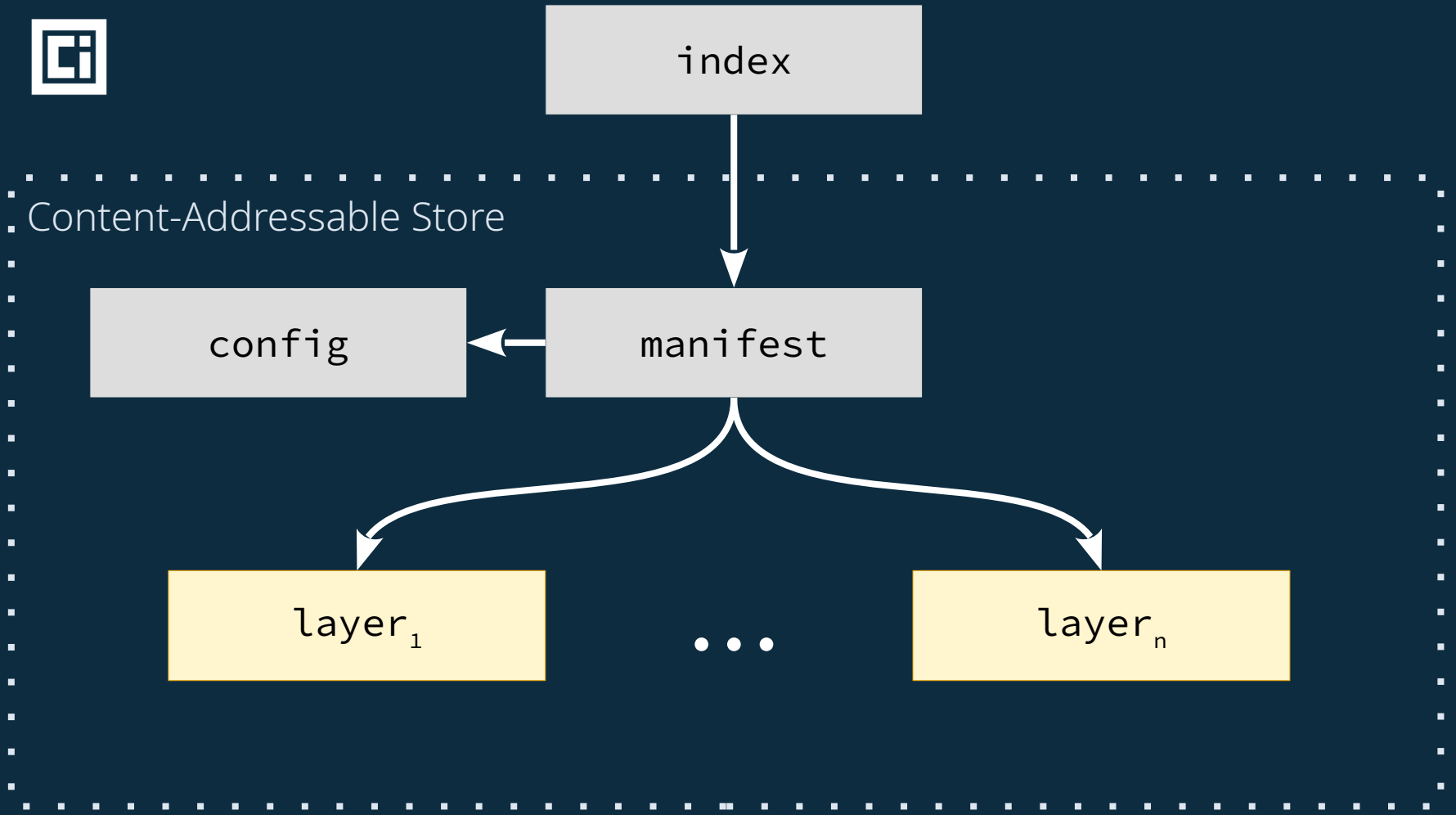

opensuse/tumbleweed1
/bin/bash
/bin/zsh
/usr/bin/ping
/usr/bin/blah
/usr/share/man/**
/usr/lib/foo.so
% oci-pull image
opensuse/tumbleweed2
/bin/bash
/bin/zsh
/usr/bin/ping
/usr/bin/blah
/usr/share/man/**
/usr/lib/foo.so
/usr/lib64/bar.so

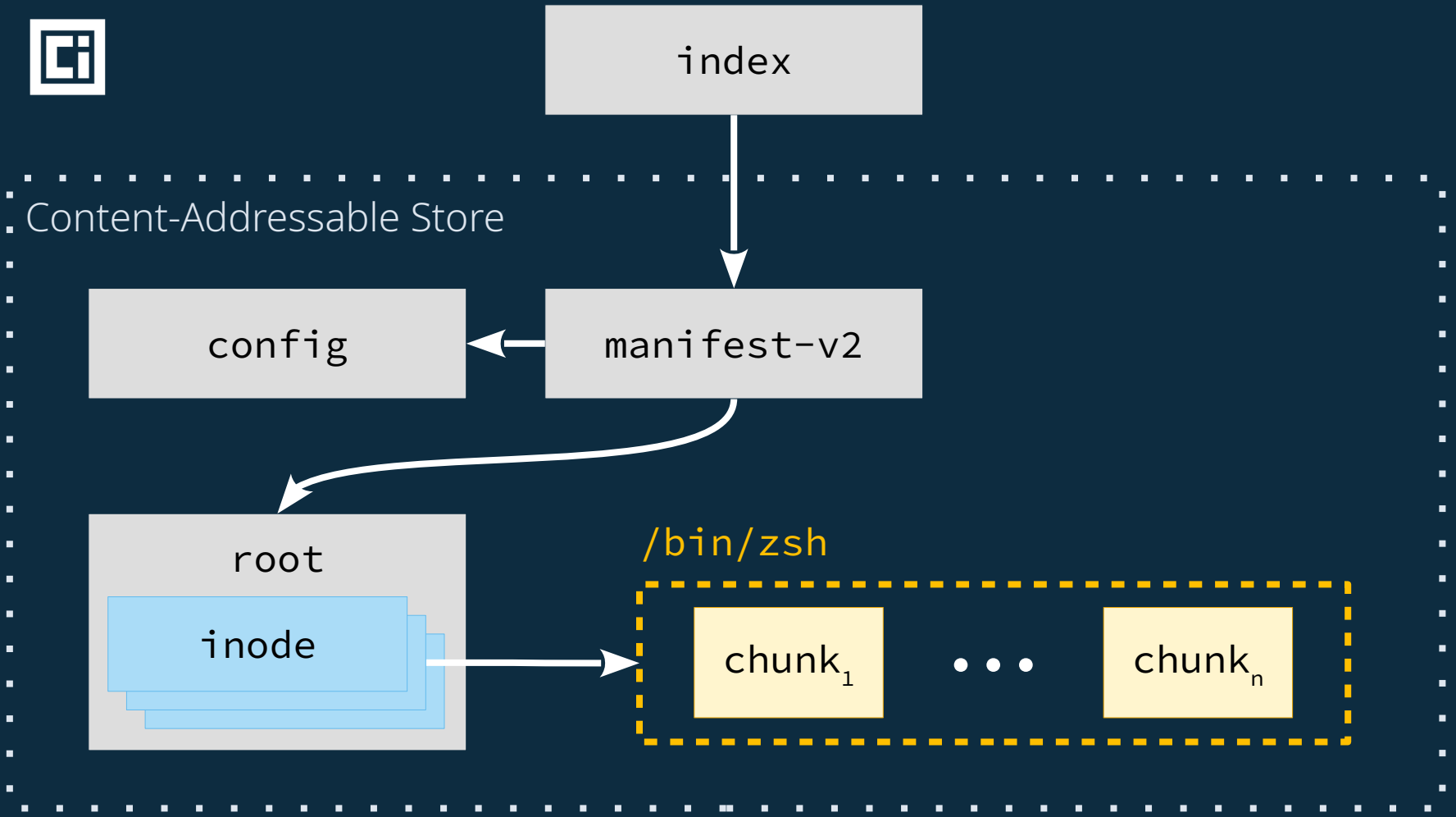

% umoci unpack
opensuse/tumbleweed1
/bin/bash
/bin/zsh
/usr/bin/ping
/usr/bin/blah
/usr/share/man/**
/usr/lib/foo.so
≠
% umoci unpack
opensuse/tumbleweed2
/bin/bash
/bin/zsh
/usr/bin/ping
/usr/bin/blah
/usr/share/man/**
/usr/lib/foo.so
/usr/lib64/bar.so

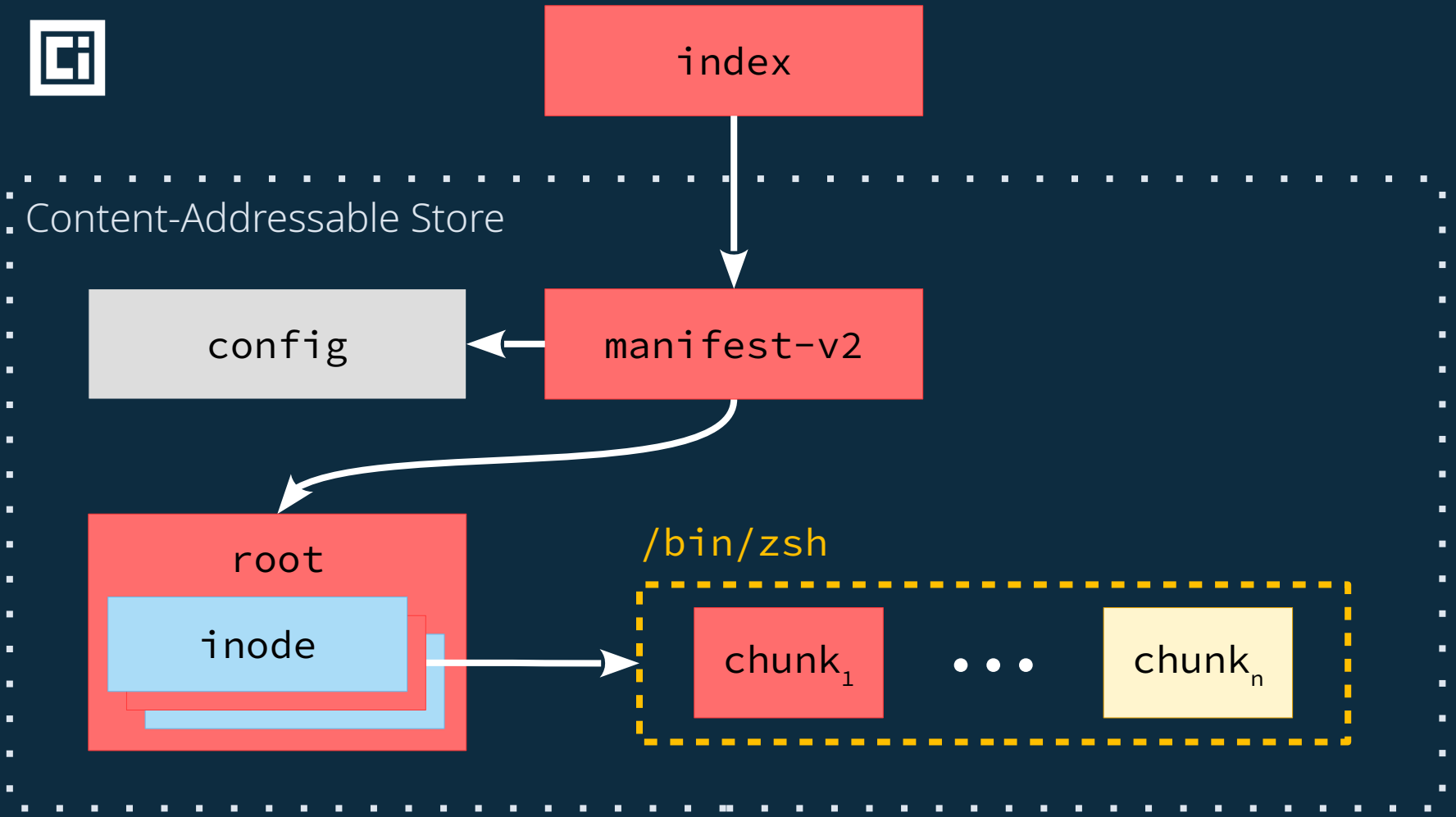

% oci-pull ubuntu
ubuntu:19.04
/bin/bash
/bin/zsh
/usr/bin/apt
/usr/bin/ping
/usr/share/man/**
/usr/lib/libc.so
% oci-pull opensuse
opensuse/tumbleweed
/bin/bash
/bin/zsh
/usr/bin/ping
/usr/bin/zypper
/usr/share/man/**
/usr/lib/libc.so

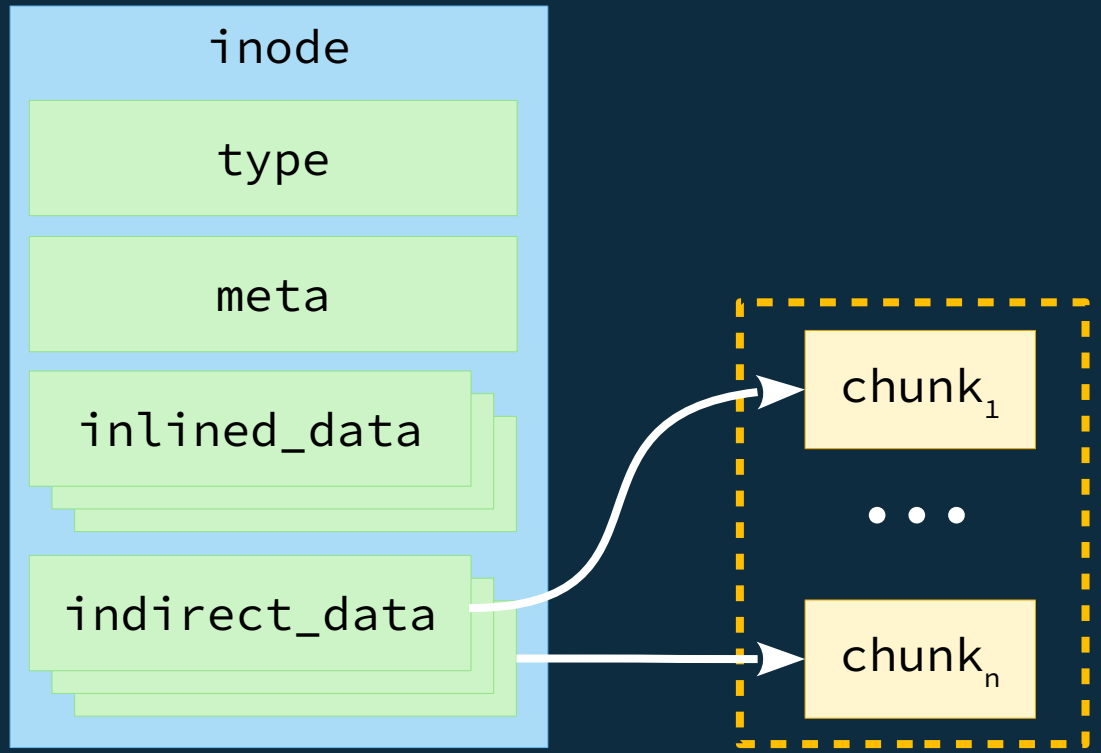

% oci-build myimg
myimg1
/foo/**
/opt/bar/**
/usr/share/man/**
/usr/lib/libc.so
≠
% oci-build myimg
myimg2
/foo/**
/usr/share/man/**
/usr/lib/libc.so
/opt/bar/**

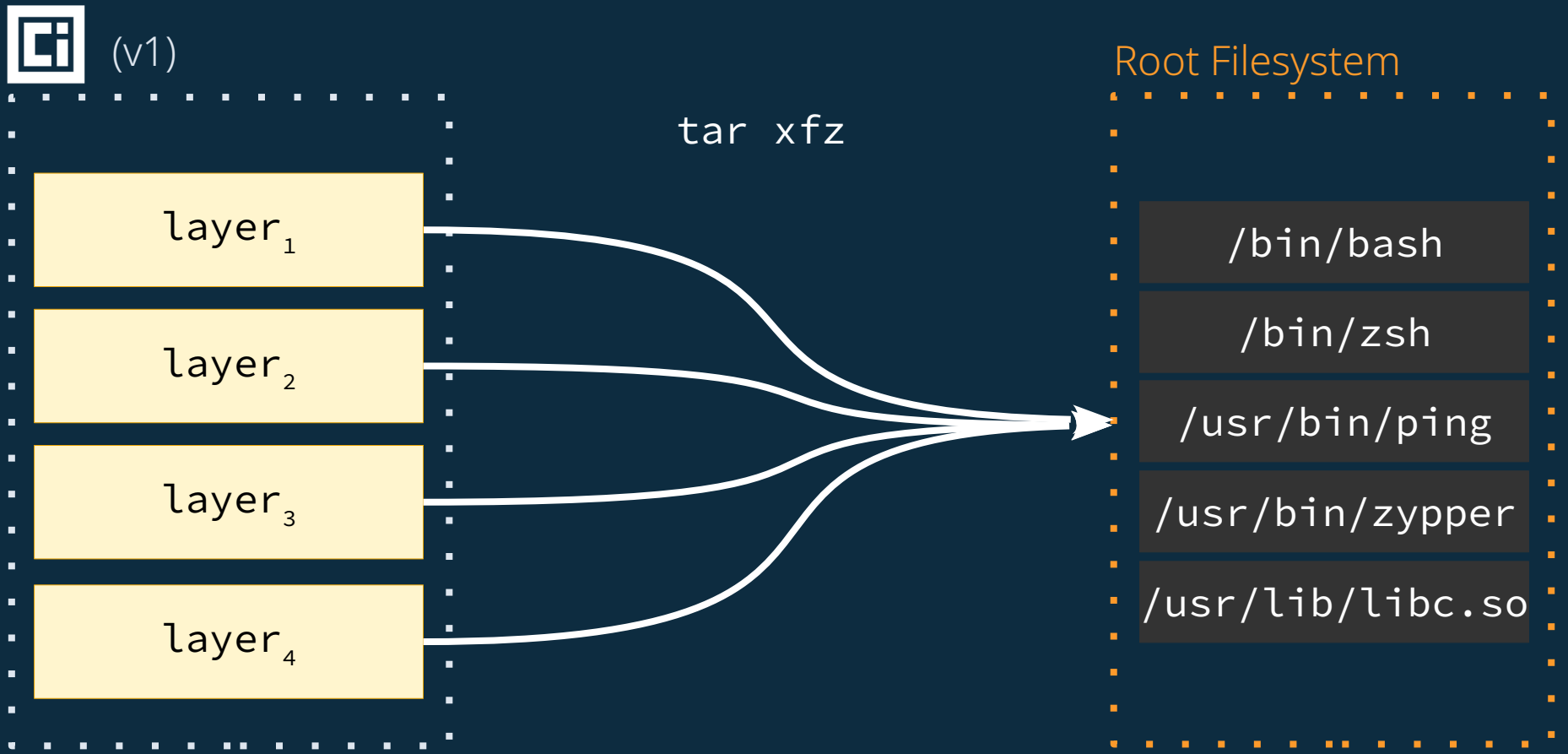
What is the alternative?

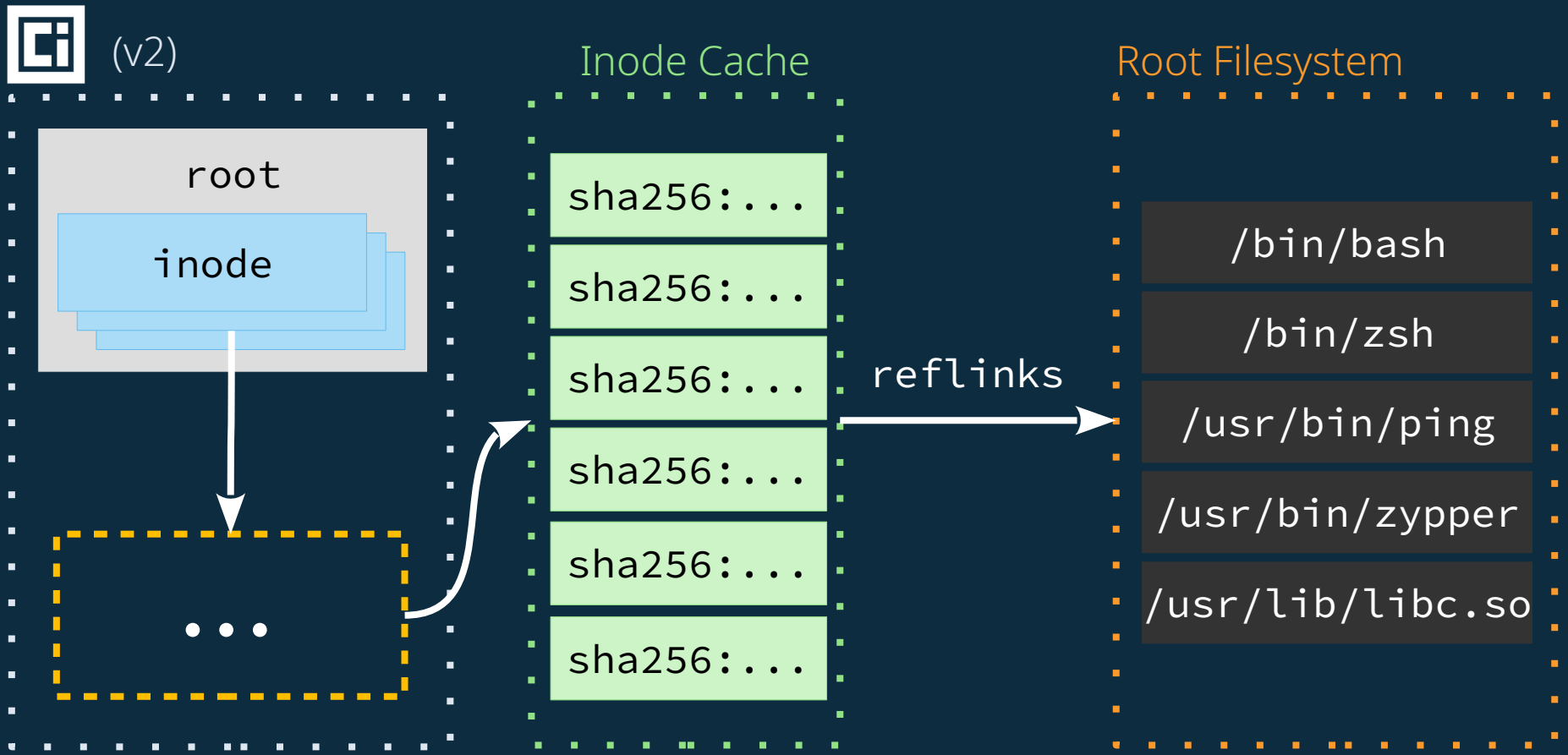

index
Content-Addressable Store
config
manifest
layer1
...
layern


index
Content-Addressable Store
config
manifest-v2
root
inode
/bin/zsh
chunk1
chunkn

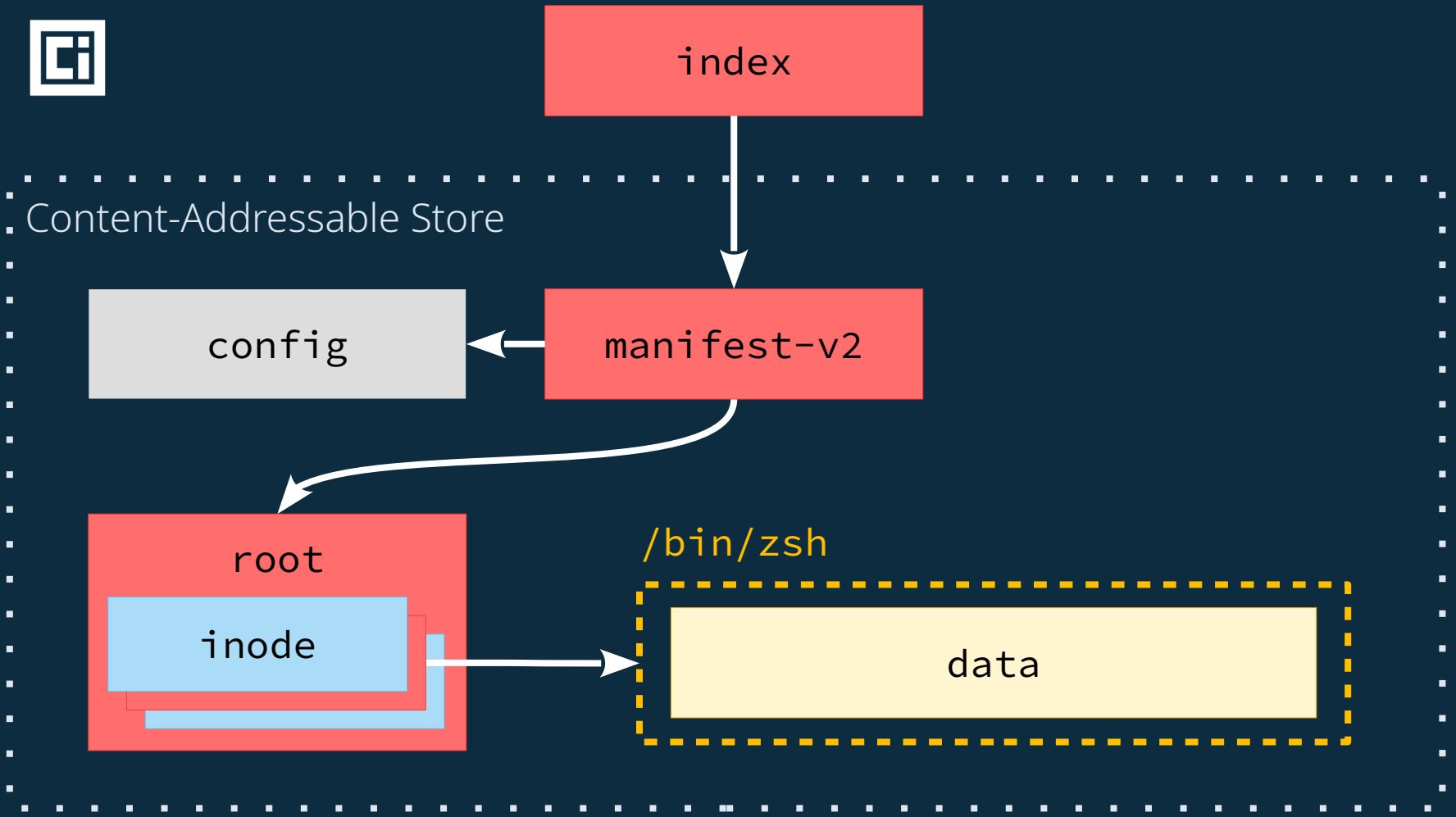

index
Content-Addressable Store
config
manifest-v2
root
inode
/bin/zsh
chunk1
chunkn

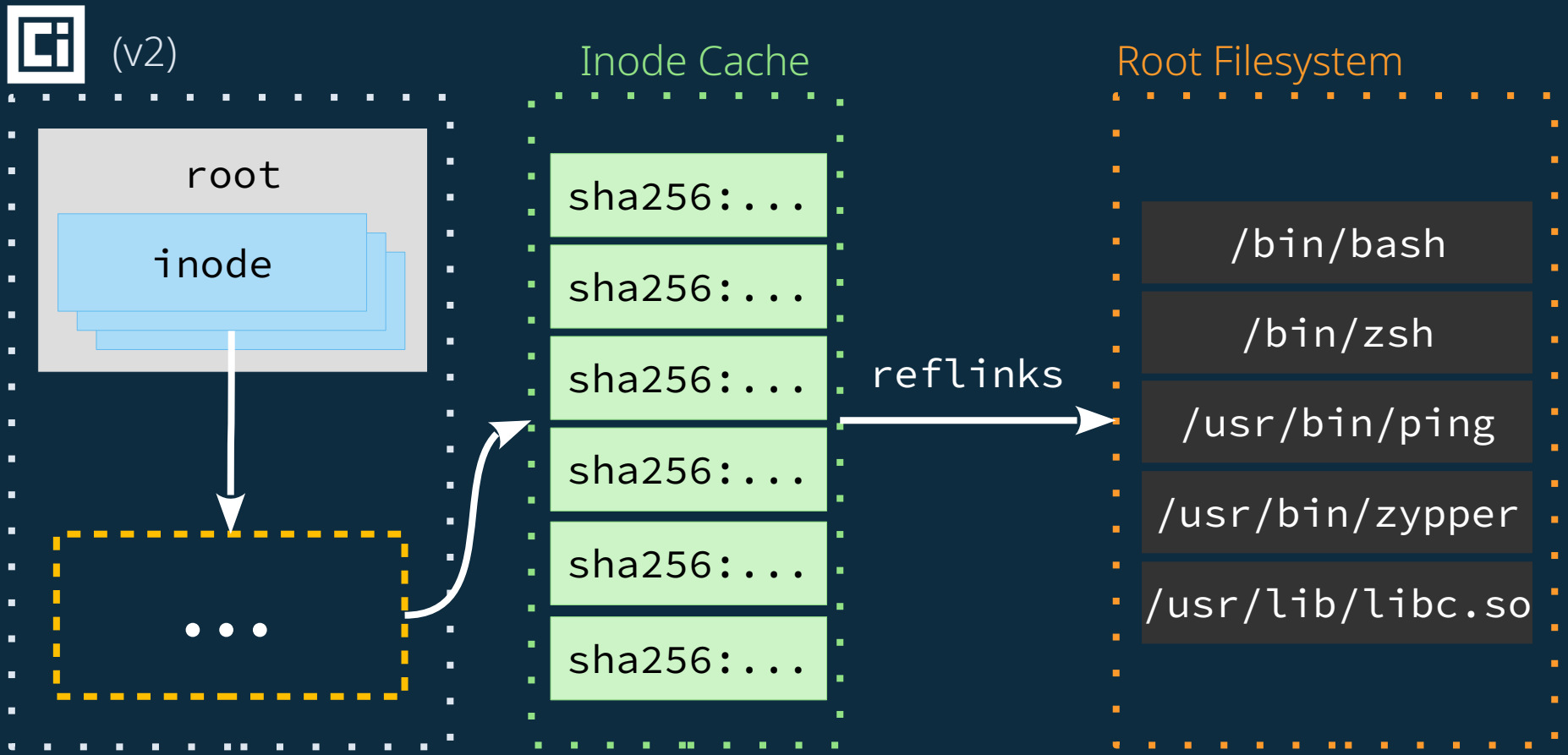

inode
type
meta
inlined_data
indirect_data
chunk1
chunkn

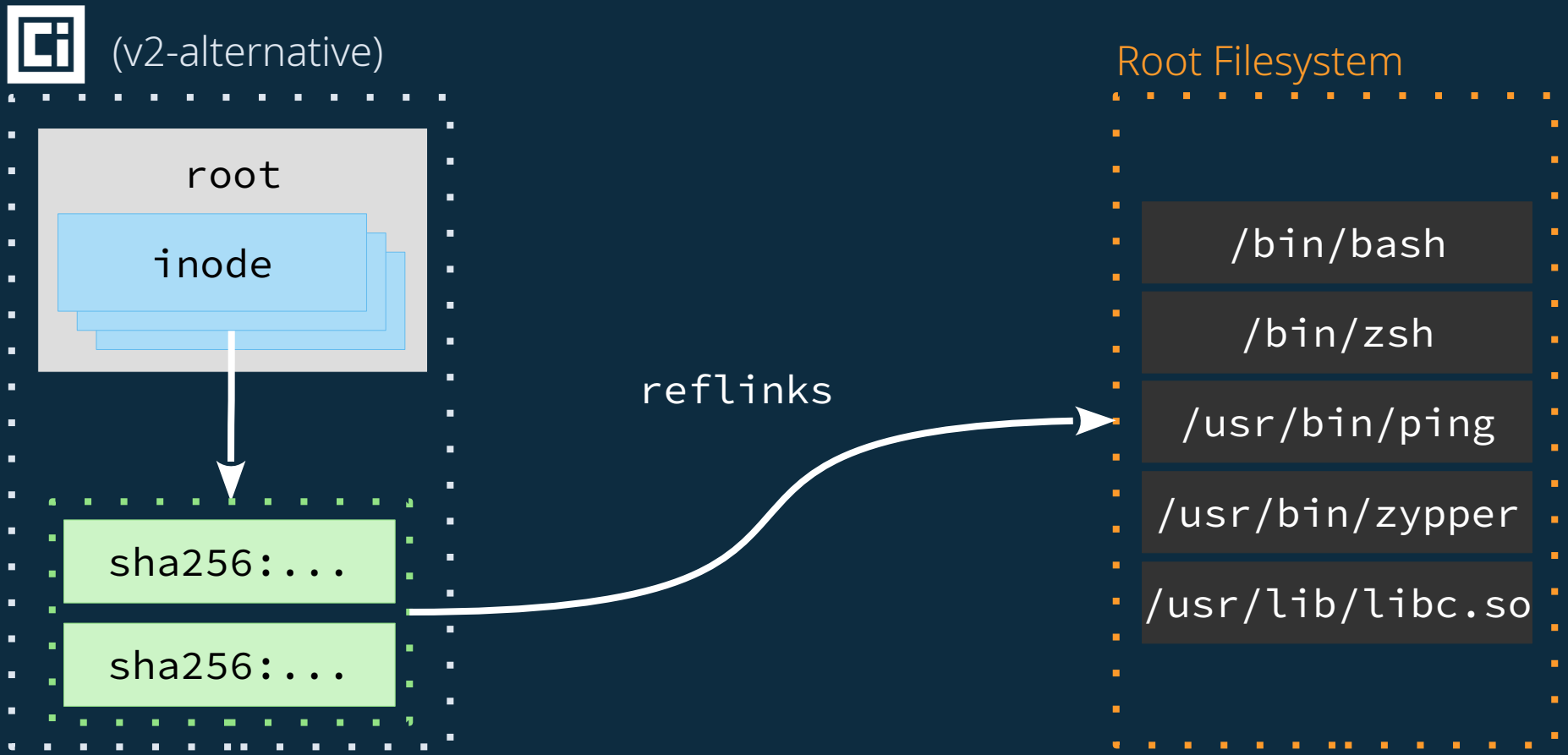

(v1)
layer1
layer2
layer3
layer4
tar xfz
Root Filesystem
/bin/bash
/bin/zsh
/usr/bin/ping
/usr/bin/zypper
/usr/lib/libc.so


(v2)
root
inode
...
Inode Cache
sha256:...
sha256:...
sha256:...
sha256:...
sha256:...
sha256:...
reflinks
Root Filesystem
/bin/bash
/bin/zsh
/usr/bin/ping
/usr/bin/zypper
/usr/lib/libc.so

Another alternative...


index
Content-Addressable Store
config
manifest-v2
root
inode
/bin/zsh
data

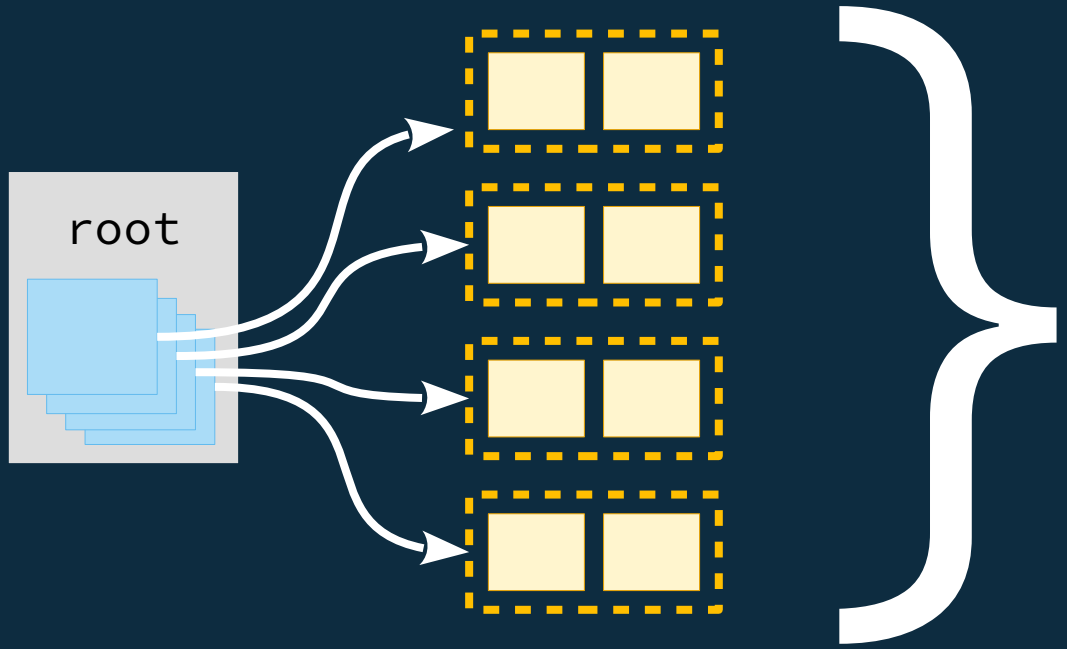

(v2)
root
inode
...
Inode Cache
sha256:...
sha256:...
sha256:...
sha256:...
sha256:...
sha256:...
reflinks
Root Filesystem
/bin/bash
/bin/zsh
/usr/bin/ping
/usr/bin/zypper
/usr/lib/libc.so


(v2-alternative)
root
inode
sha256:...
sha256:...
reflinks
Root Filesystem
/bin/bash
/bin/zsh
/usr/bin/ping
/usr/bin/zypper
/usr/lib/libc.so

Stop!

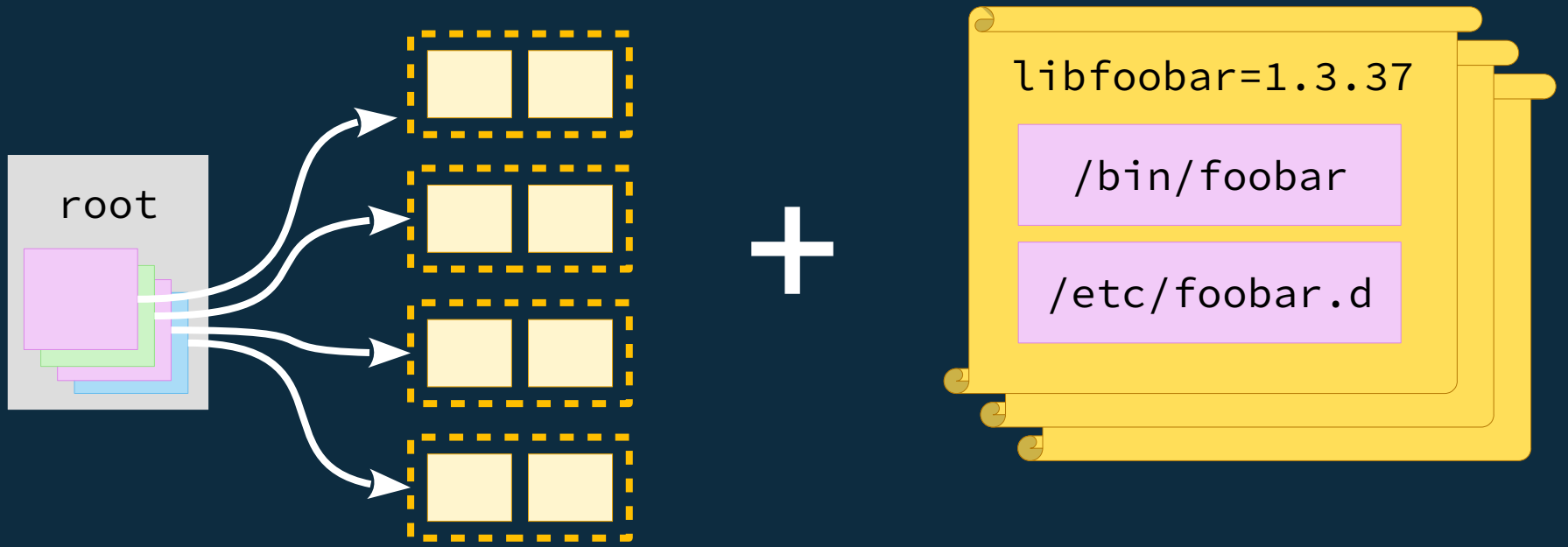
Demo time.

Bill of Materials



What is actually in this thing?

Bill of Materials



Yet More Alternatives...

- `catar` (from systemd) does attempt to solve similar problems.
  - Might be useful for transfer, but doesn't match our reflink storage needs.
  - Maybe using a "single blob" model for transfer (like `catar`) would help.
    - OCI distribution could use HTTP Range requests.
    - Have a catar-like jump-table at the tail of the blob.

Where?

- https://github.com/openSUSE/umoci
  - There is a ***very*** experimental branch with the demo code.
- https://www.cyphar.com/blog/post/20190121-ociv2-images-i-tar/
  - The much more long-form rant about `tar`.
- https://github.com/cyphar/talks
  - Where you can get the slides later.

Next Steps?

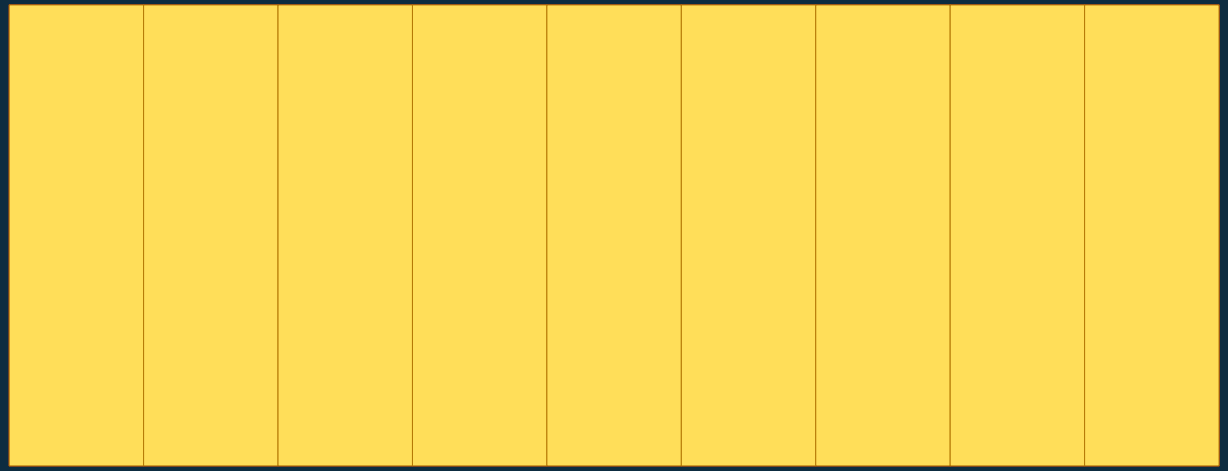
- Reduce size of transfers (compression still beats us in many cases).
- Design the “bill of materials” format.
- Write a specification and submit it for review.
  - Make sure all possible users are happy with switching.
- Get everyone to switch.
  - This last one might take a while.

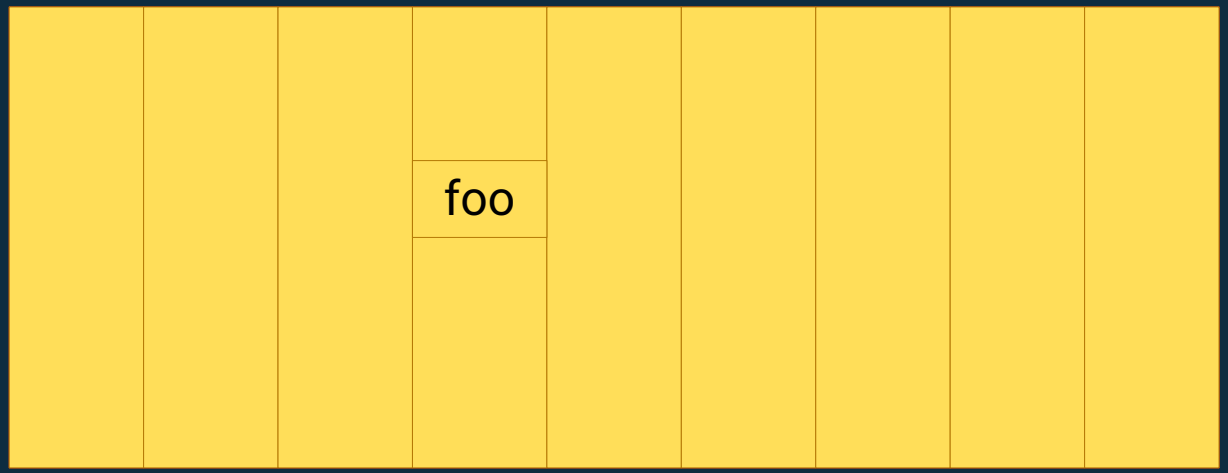
Where?

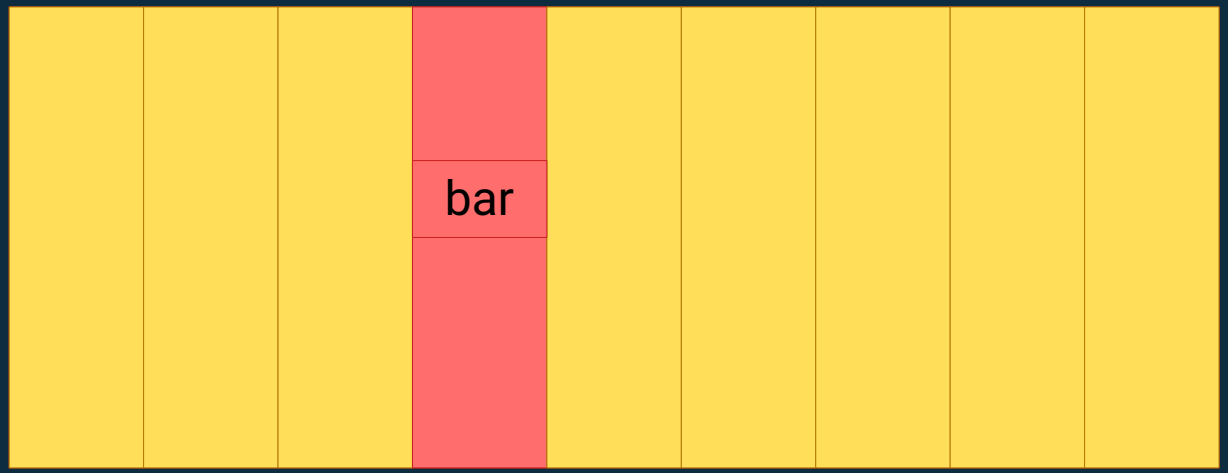
- https://github.com/openSUSE/umoci
  - There is a ***very*** experimental branch with the demo code.
- https://www.cyphar.com/blog/post/20190121-ociv2-images-i-tar/
  - The much more long-form rant about `tar`.
- https://github.com/cyphar/talks
  - Where you can get the slides later.

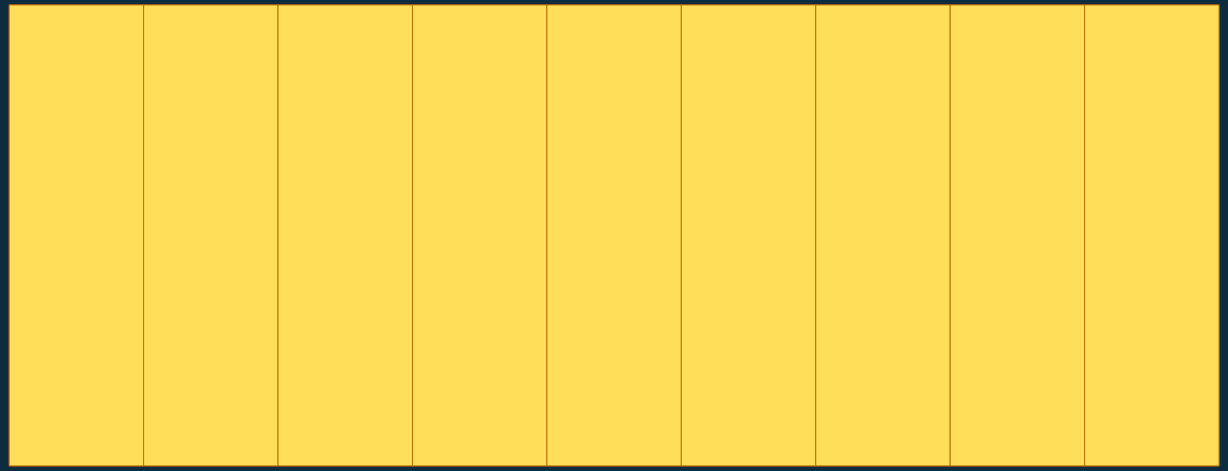
Questions?

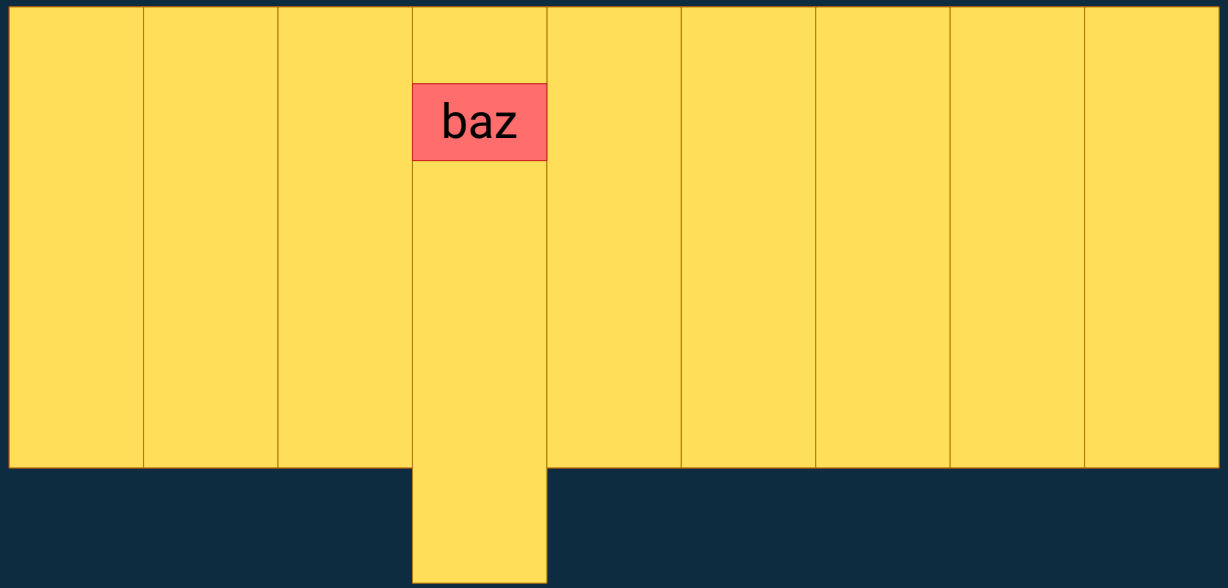
Pictured: A Humble File.



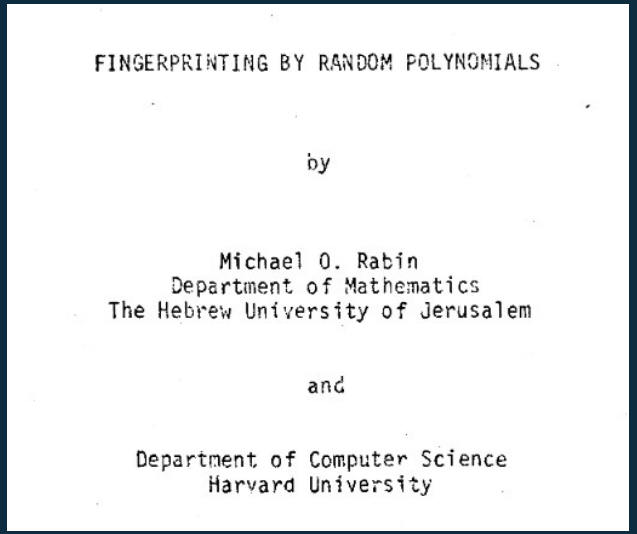

foo


bar




baz


baz



FINGERPRINTING BY RANDOM POLYNOMIALS

by

Michael O. Rabin
Department of Mathematics
The Hebrew University of Jerusalem

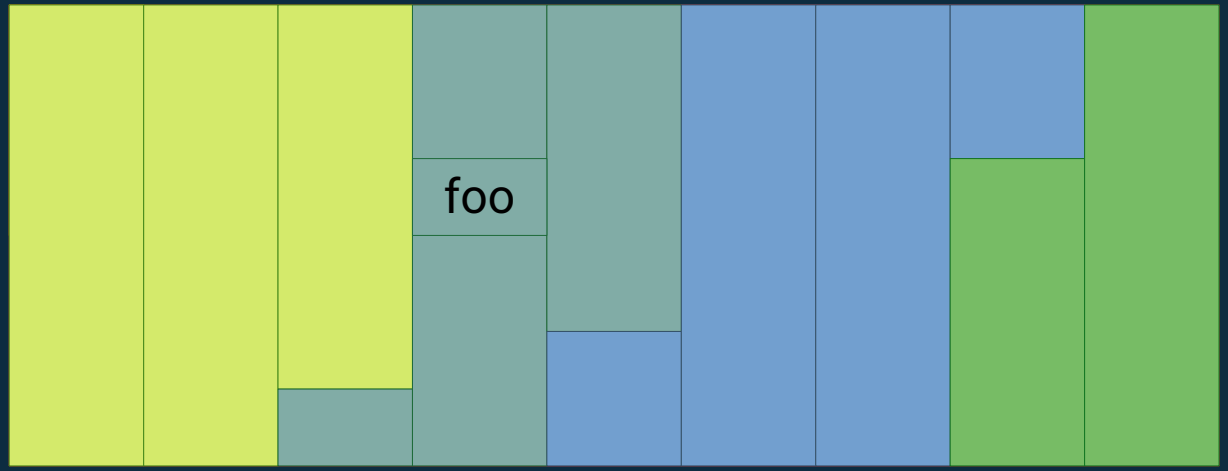
and

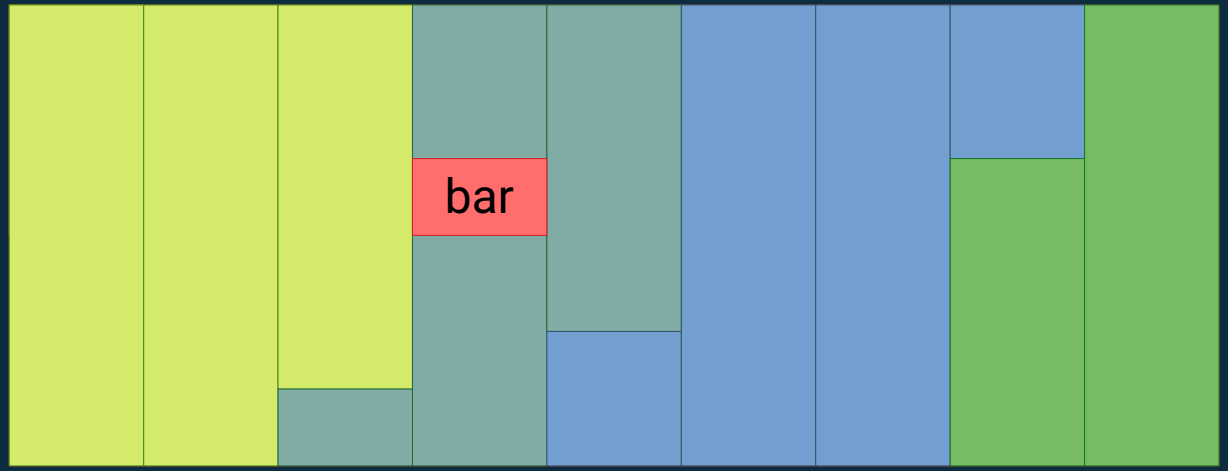
Department of Computer Science
Harvard University

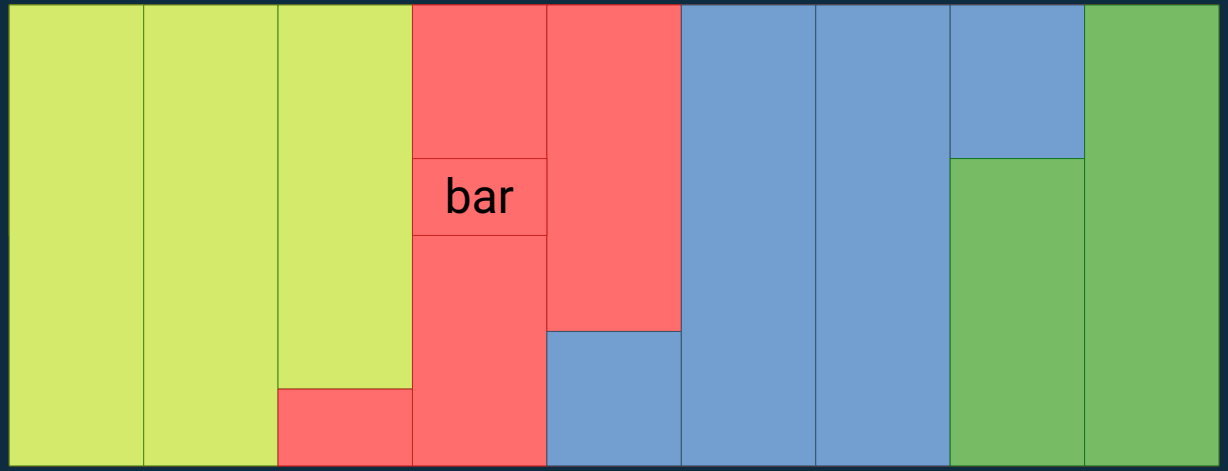
We have to go back (to 1981)!

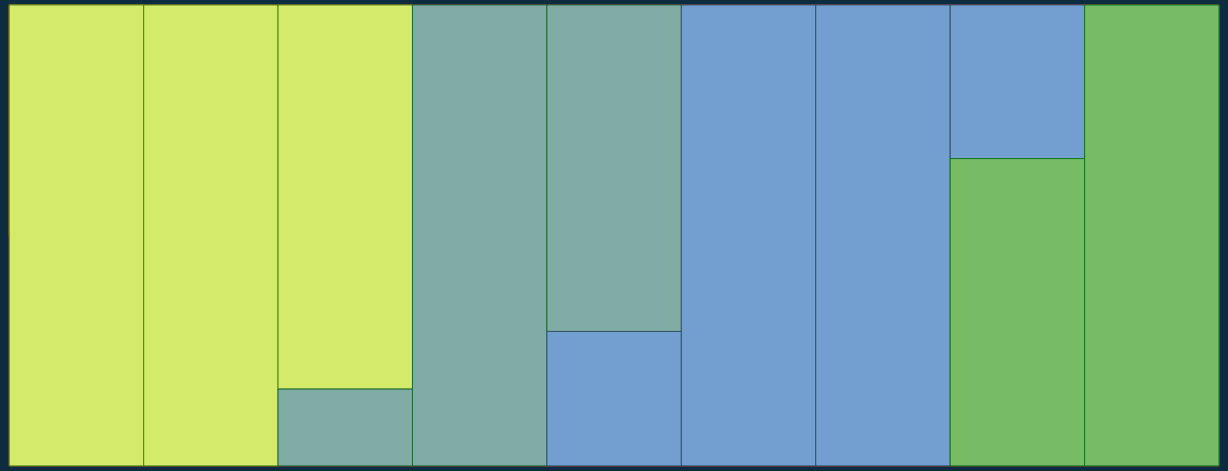
Content-Defined Chunking

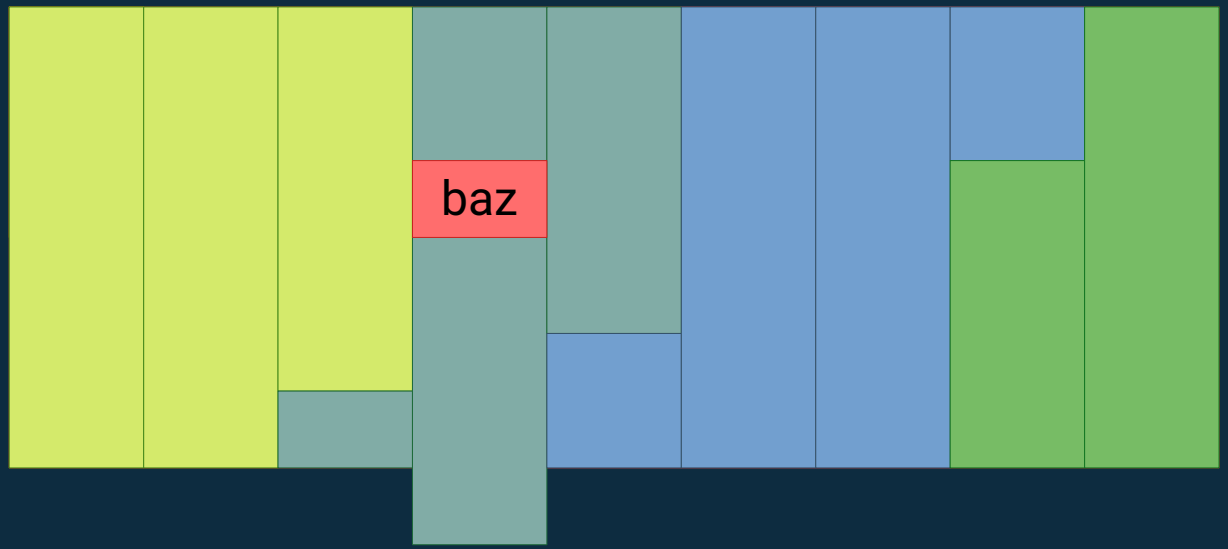
Pictured: A(nother) Humble File.

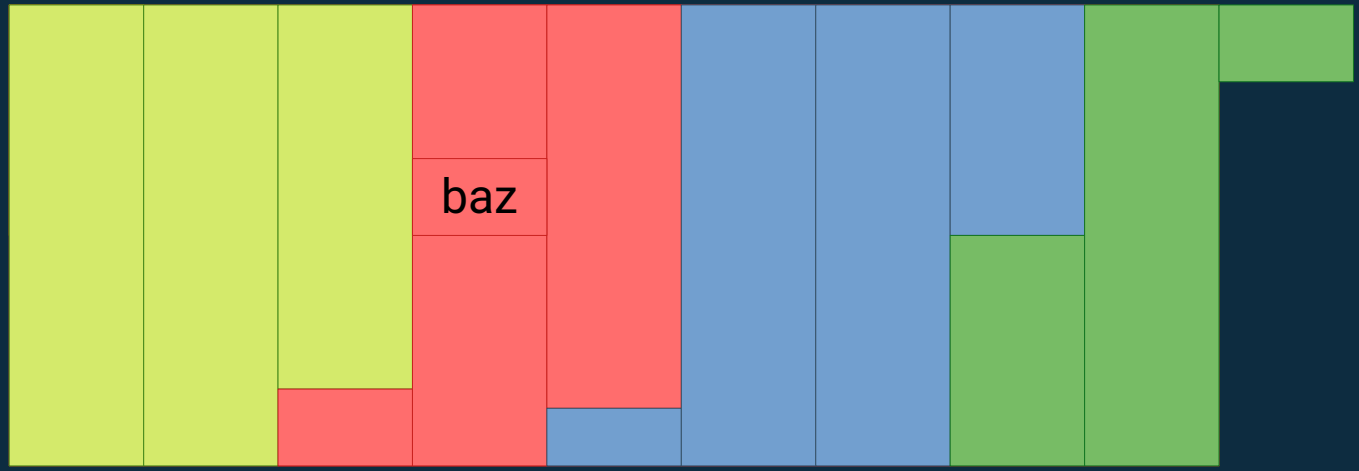

foo


bar


bar




baz


baz