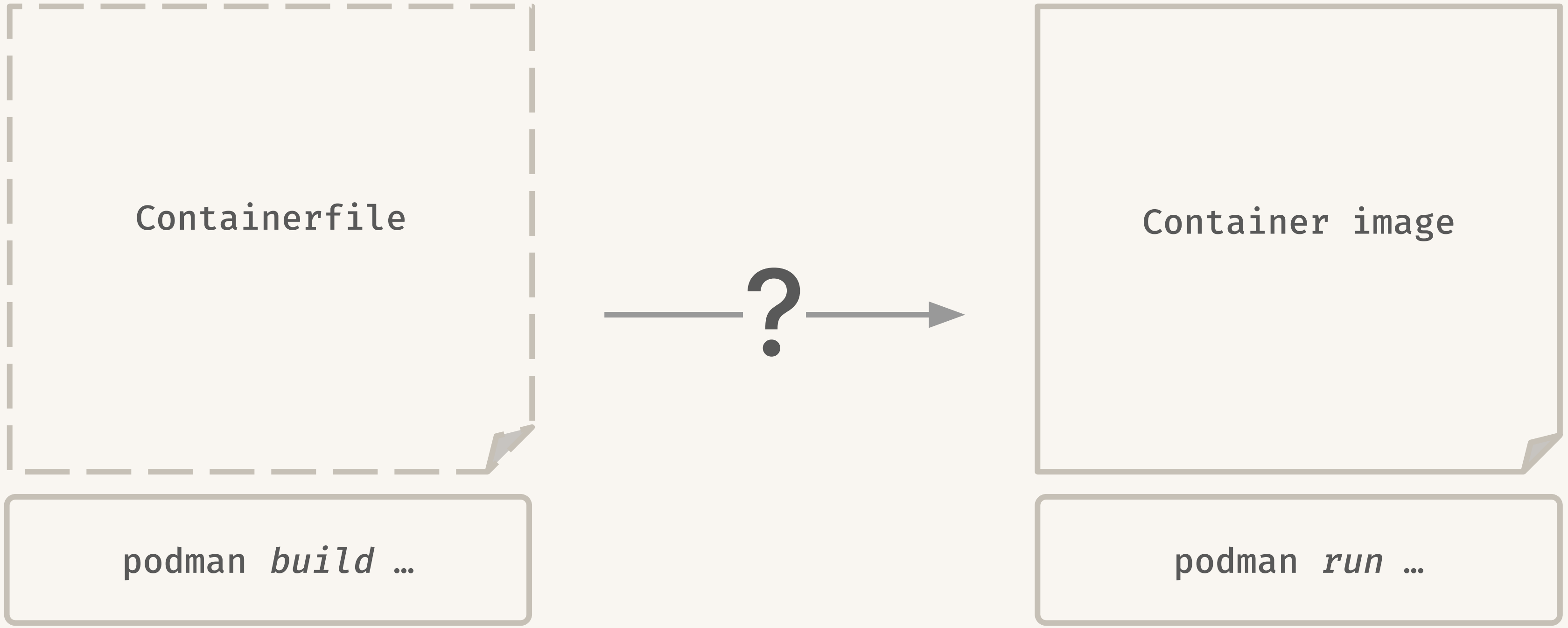
Build a Container Image from Scratch

danishpraka.sh


Containerfile
?
Container image
podman *build* …
podman *run* …

container image?

what makes up a container image?

let’s build an image

let’s build an image based on our learnings

demo & verification

demo, load and inspect the image via podman

container image?

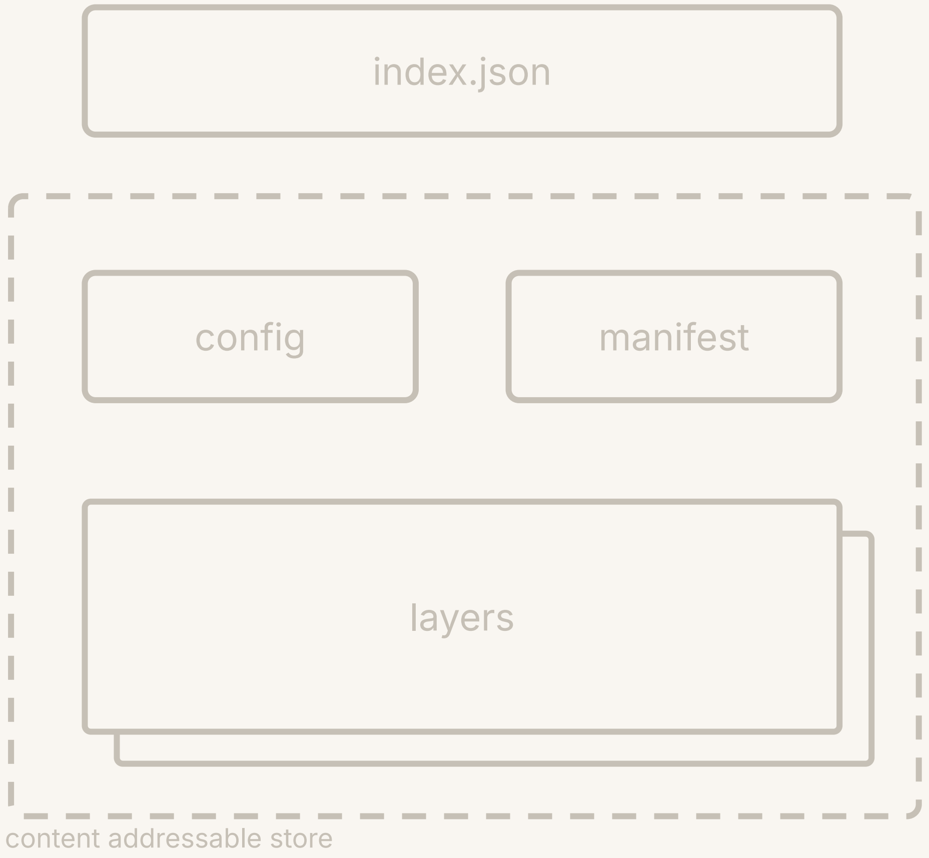
OCI image

what constitutes an OCI image

→ Open Containers Initiative

→ Image Specification

→ four major components



1. layer

what's *inside* the container image

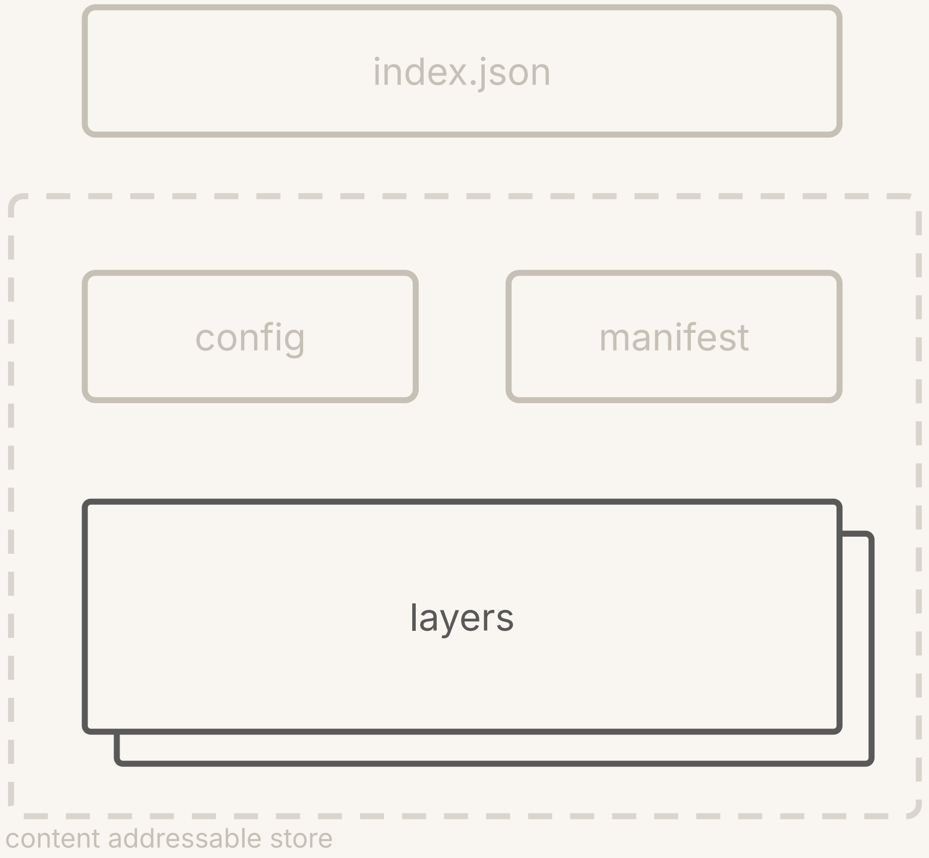
building block for containers

→ filesystem

→ source code

filesystem changeset

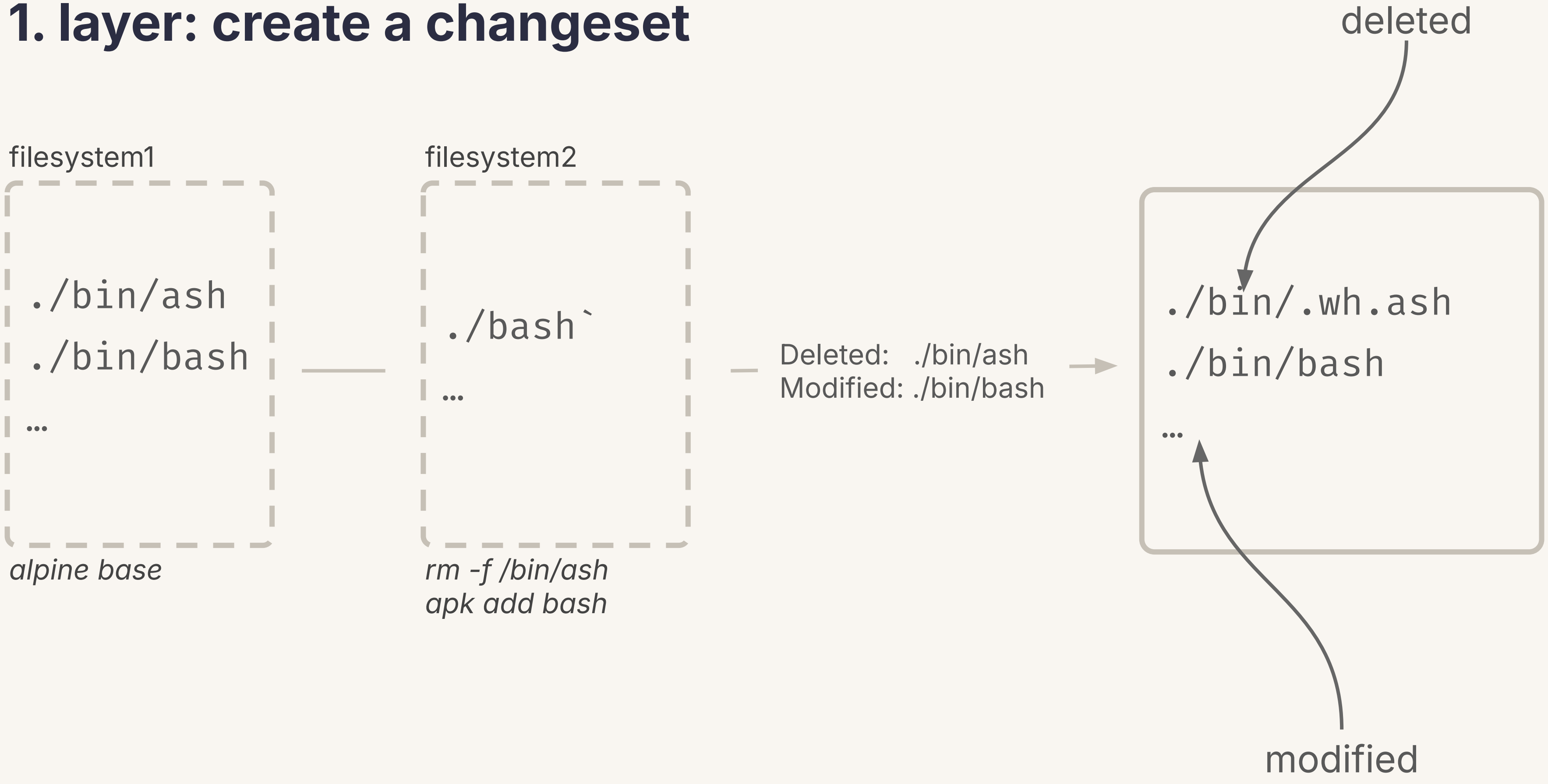
→ diff rootfs1..rootfs2



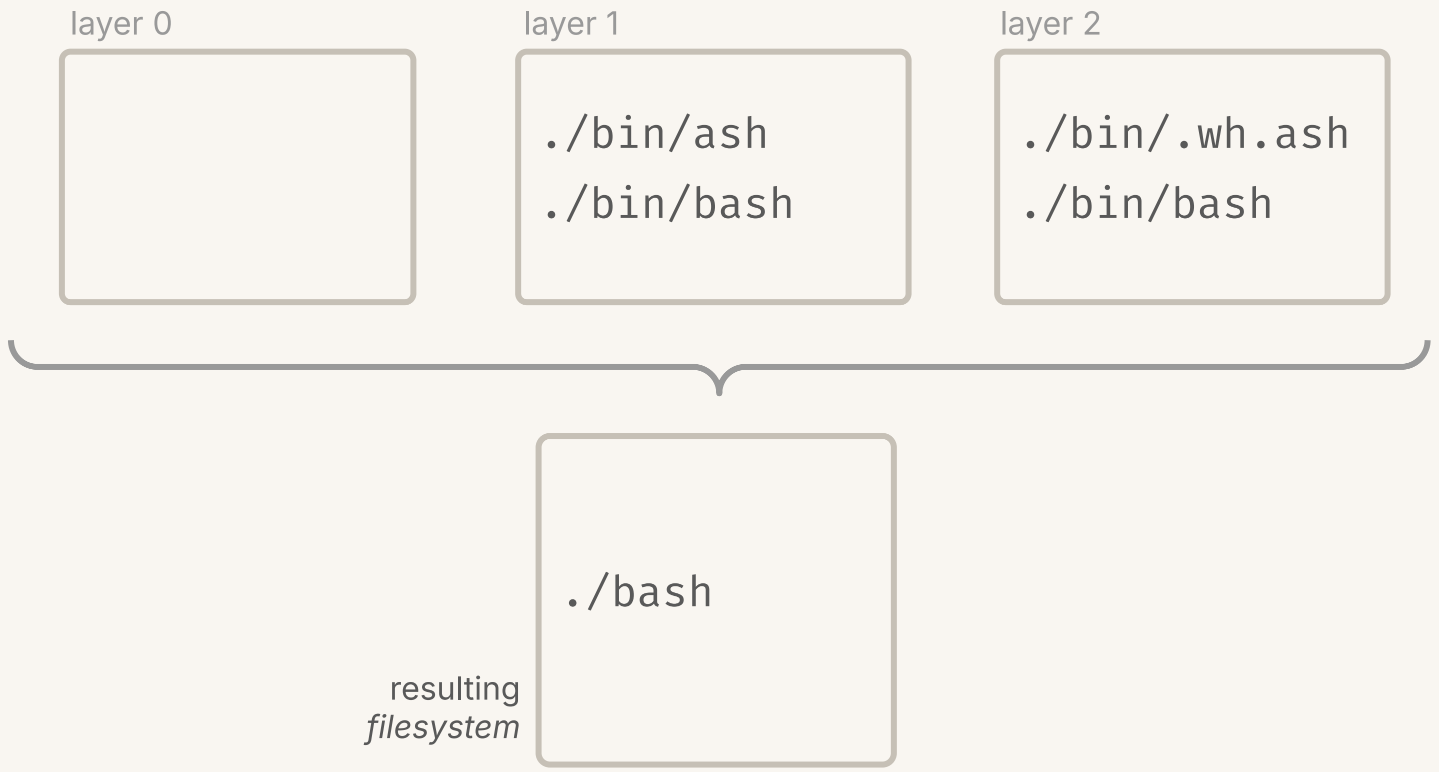
1. layer: create a changeset

```
FROM alpine

RUN rm -f /bin/ash \ # delete ash
    && apk add bash  # update bash

…
```

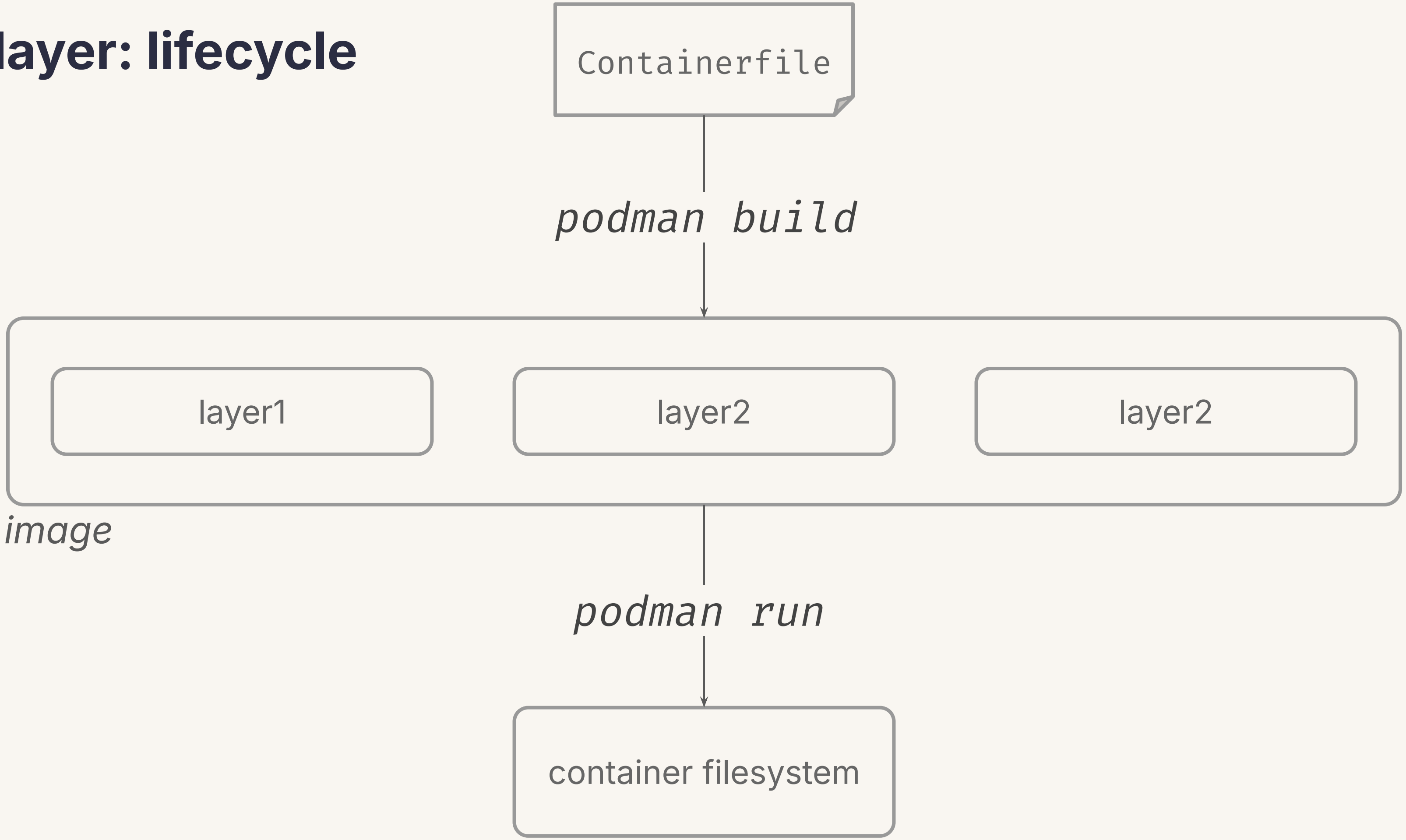
1. layer: create a changeset



1. layer: create the filesystem



1. layer: lifecycle



2. config

how to *run* the container

environment variables?

→ HOSTNAME=globalhost

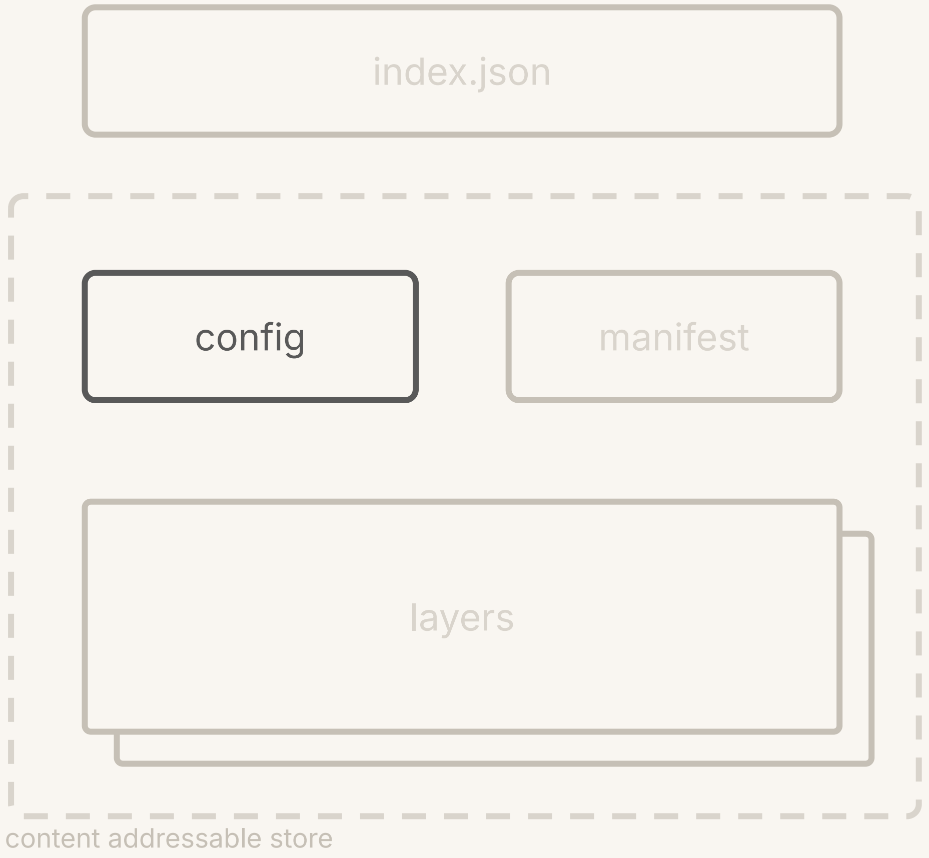
→ PATH=/bin

container entrypoint

→ ./start.sh

container cli options

```
→ podman run \
    --volume
    --env
    --port
```



3. manifest

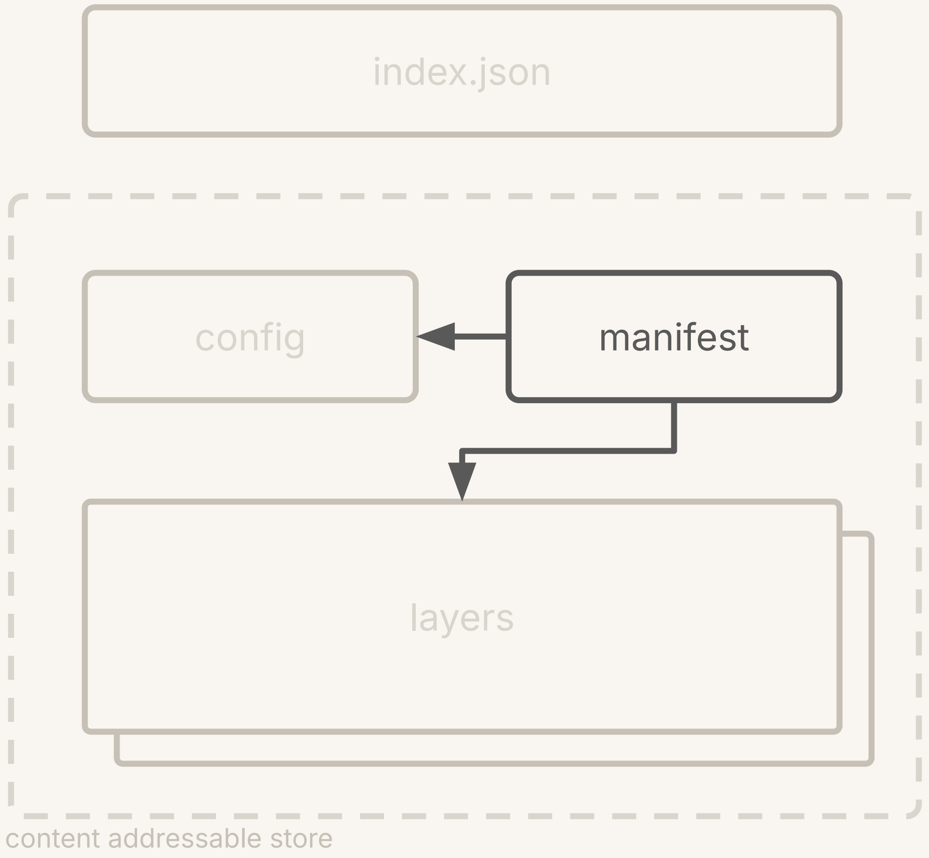
provide config *and* layer info

configuration digest

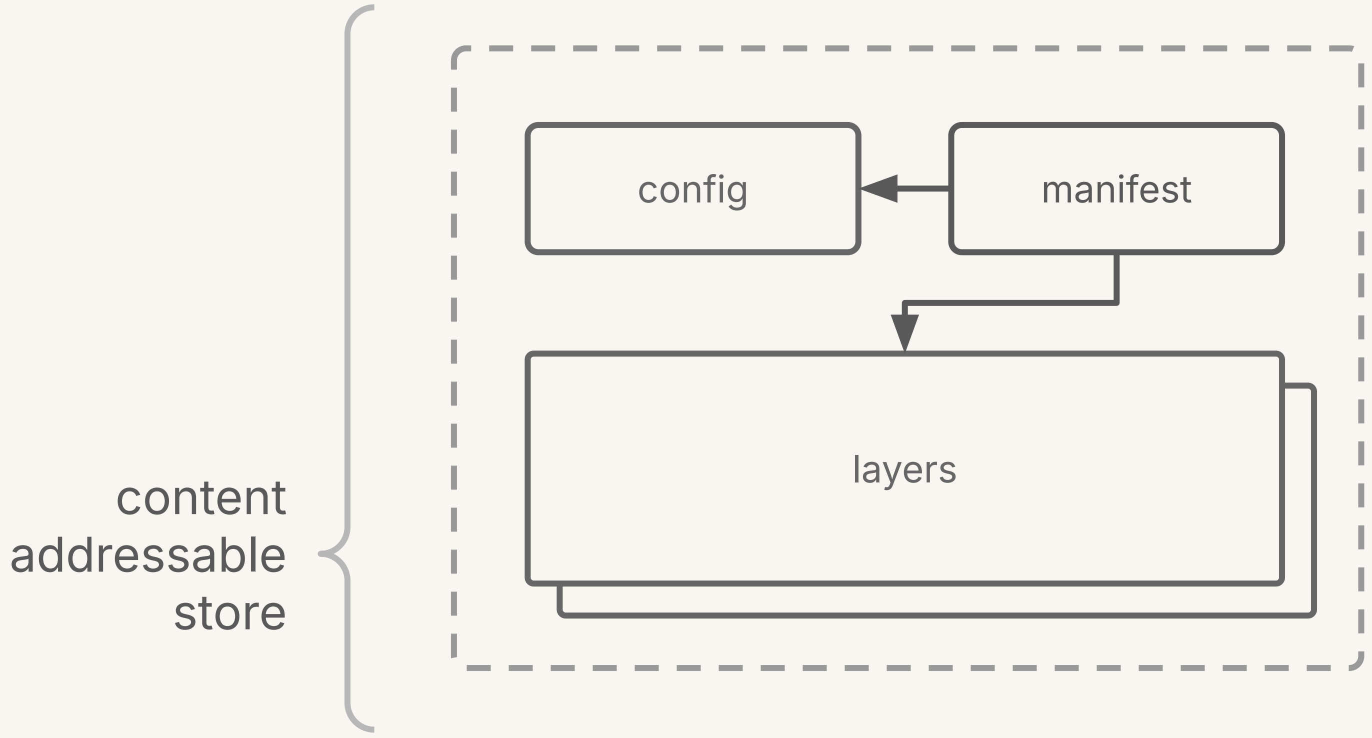
→ sha256:75b148a9a5b

layer digests

→ sha256:c37c06cdec9



content addressable store

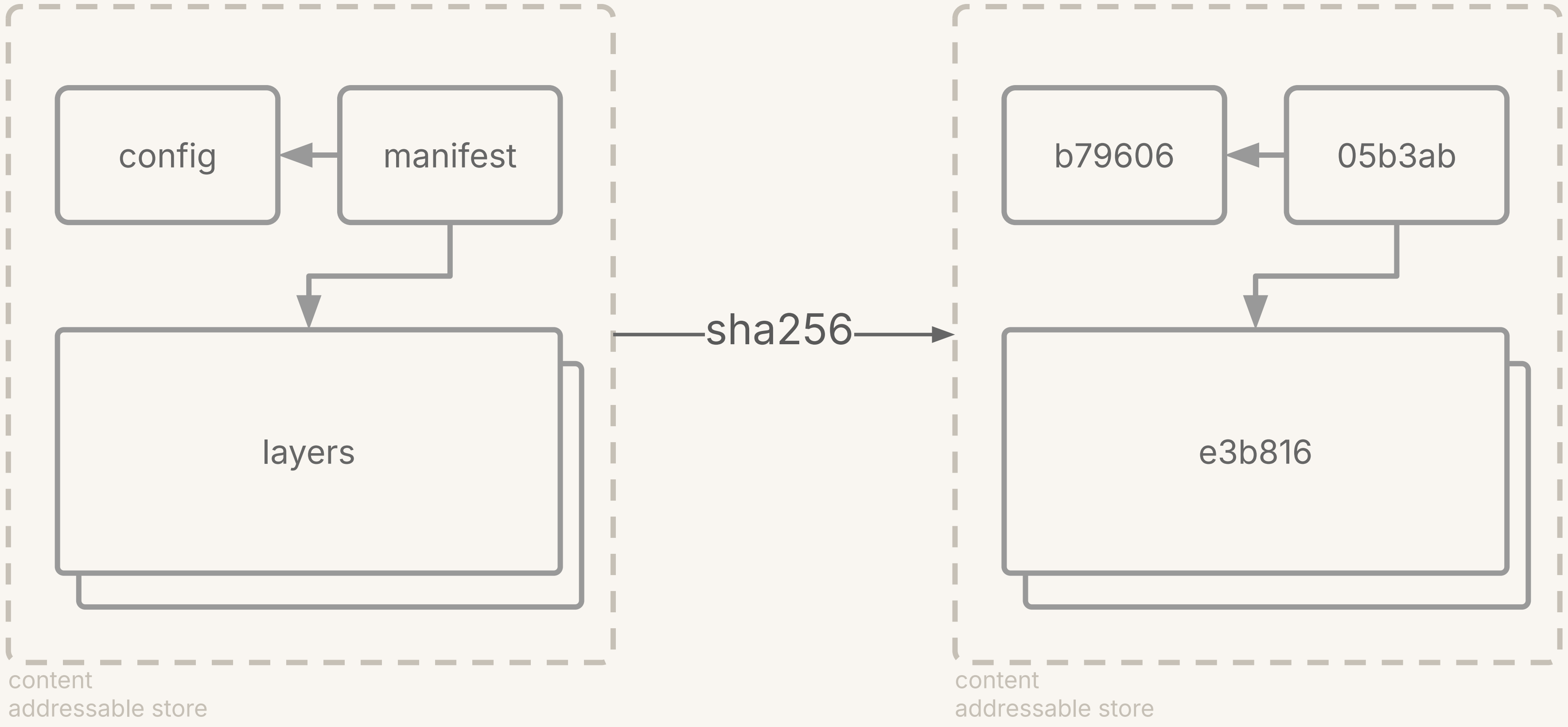


content addressability

```
$ cat event.md
Open Source Summit

$ sha256sum event.md
4c51d469b15d3423dd2d2…

$ mv event.md 4c51d469b15d3423dd2d2…
$ cat 4c51d469b15d3423dd2d2…
Open Source Summit
```

content address

content addressable store



4. index

manifest *for* manifest.json(s)

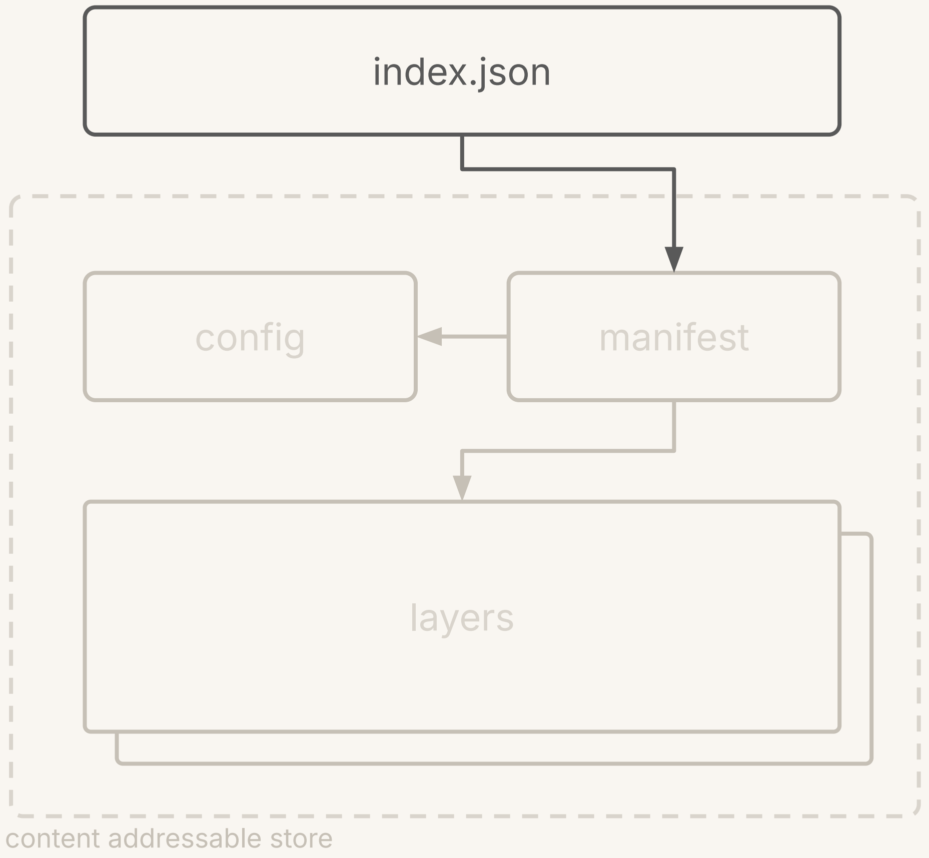
points to manifest.json

→ sha256:8289bd1bdc2

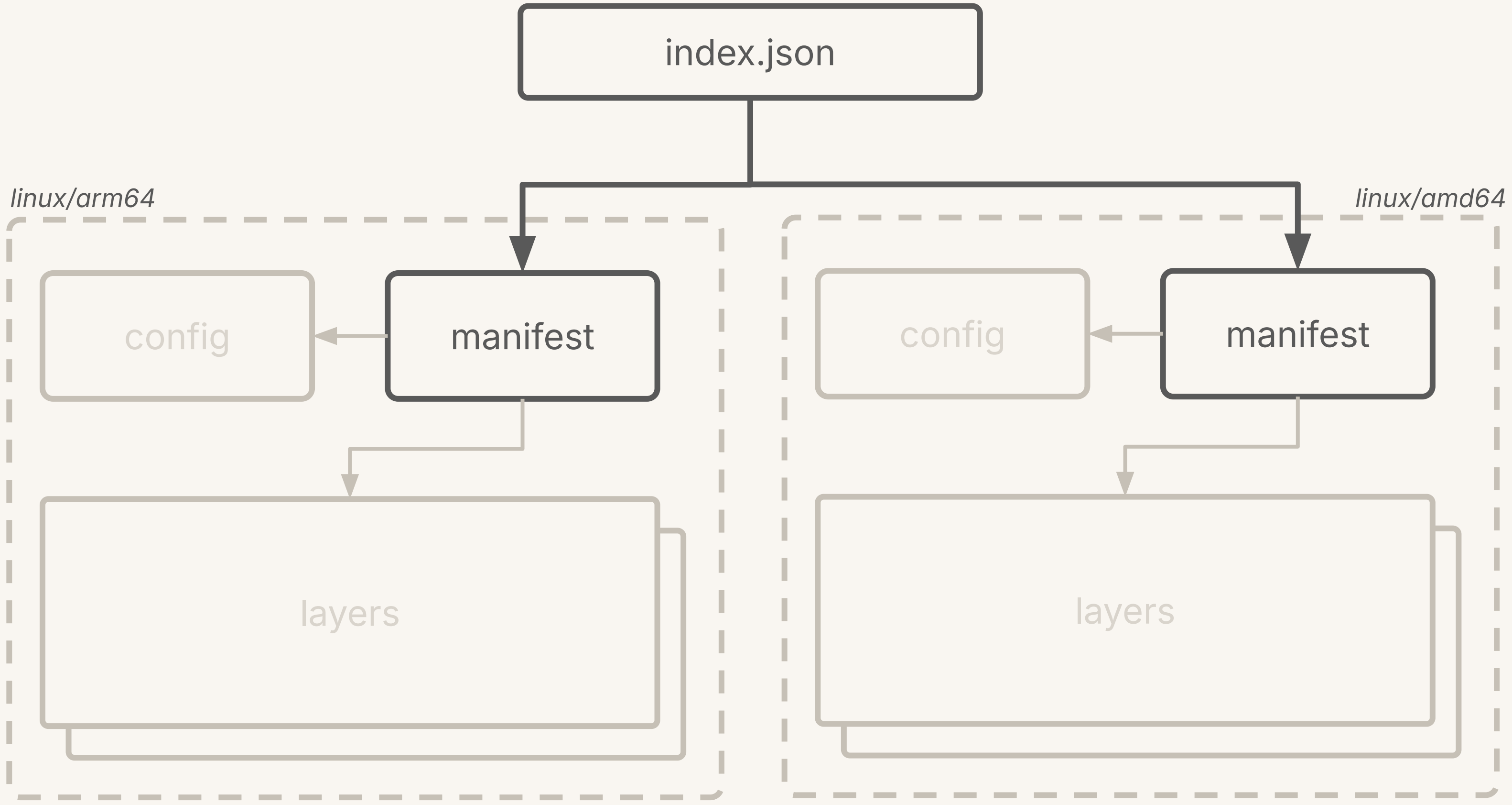
spans different platforms

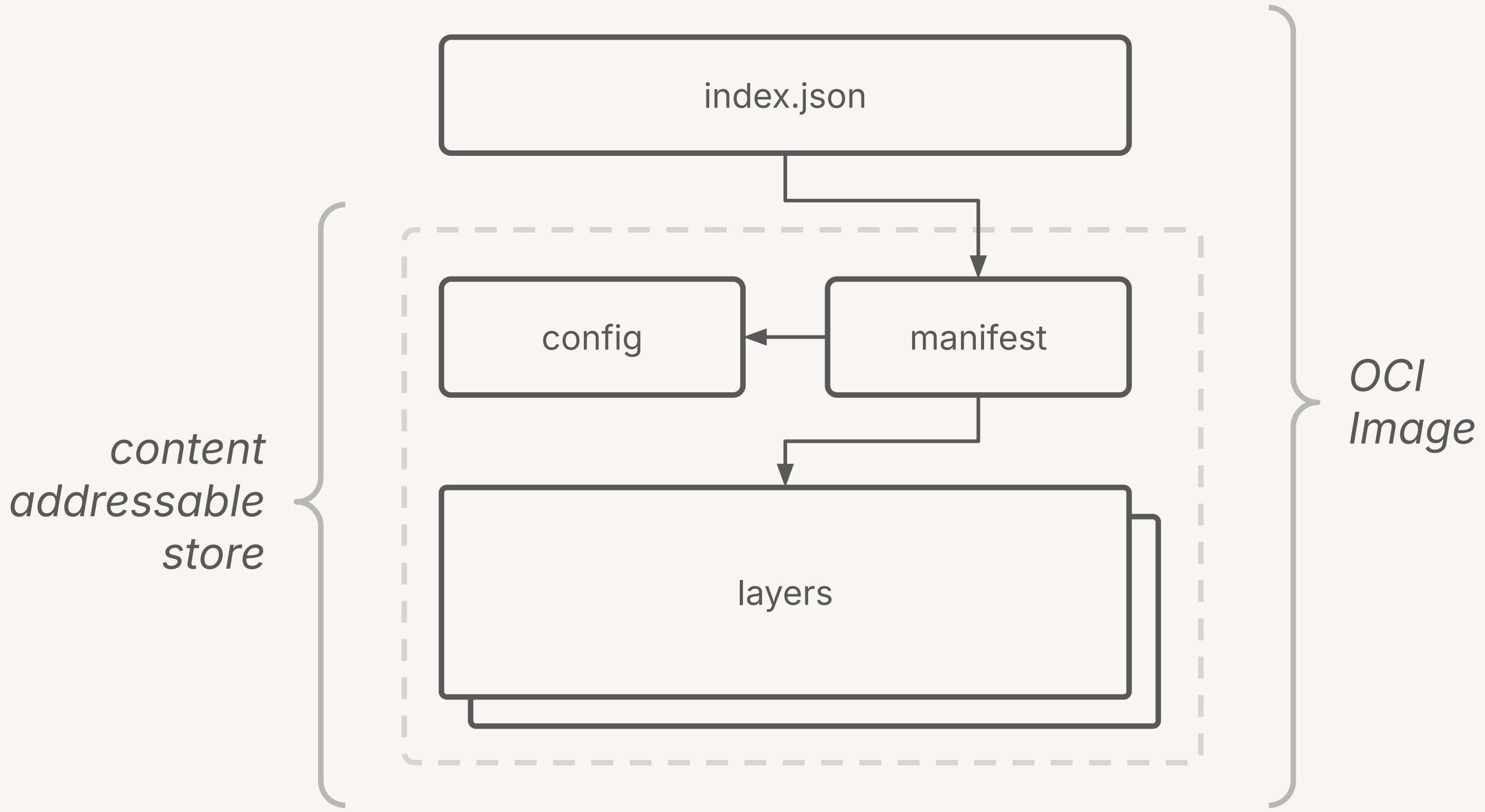
→ linux/amd64

→ linux/arm



multi-platform

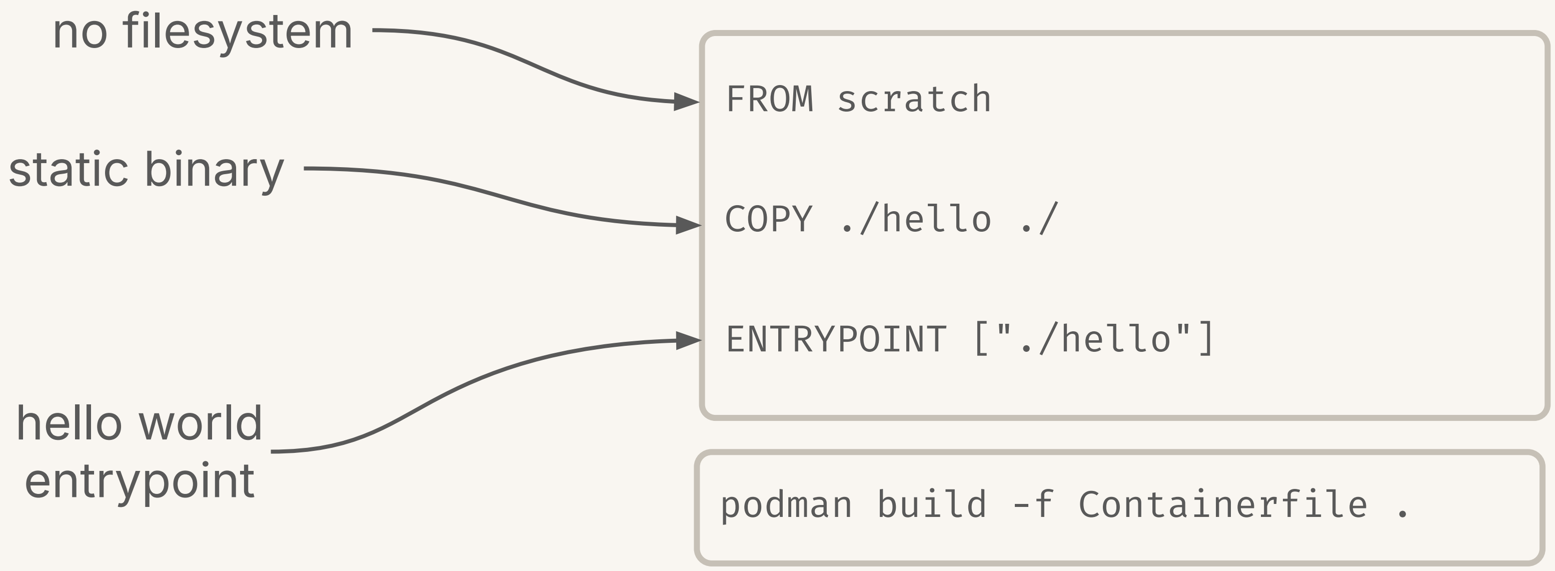



index.json
config
manifest
layers
content addressable store
OCI Image

let’s build an image!

scratch image

first variant with *no* filesystem

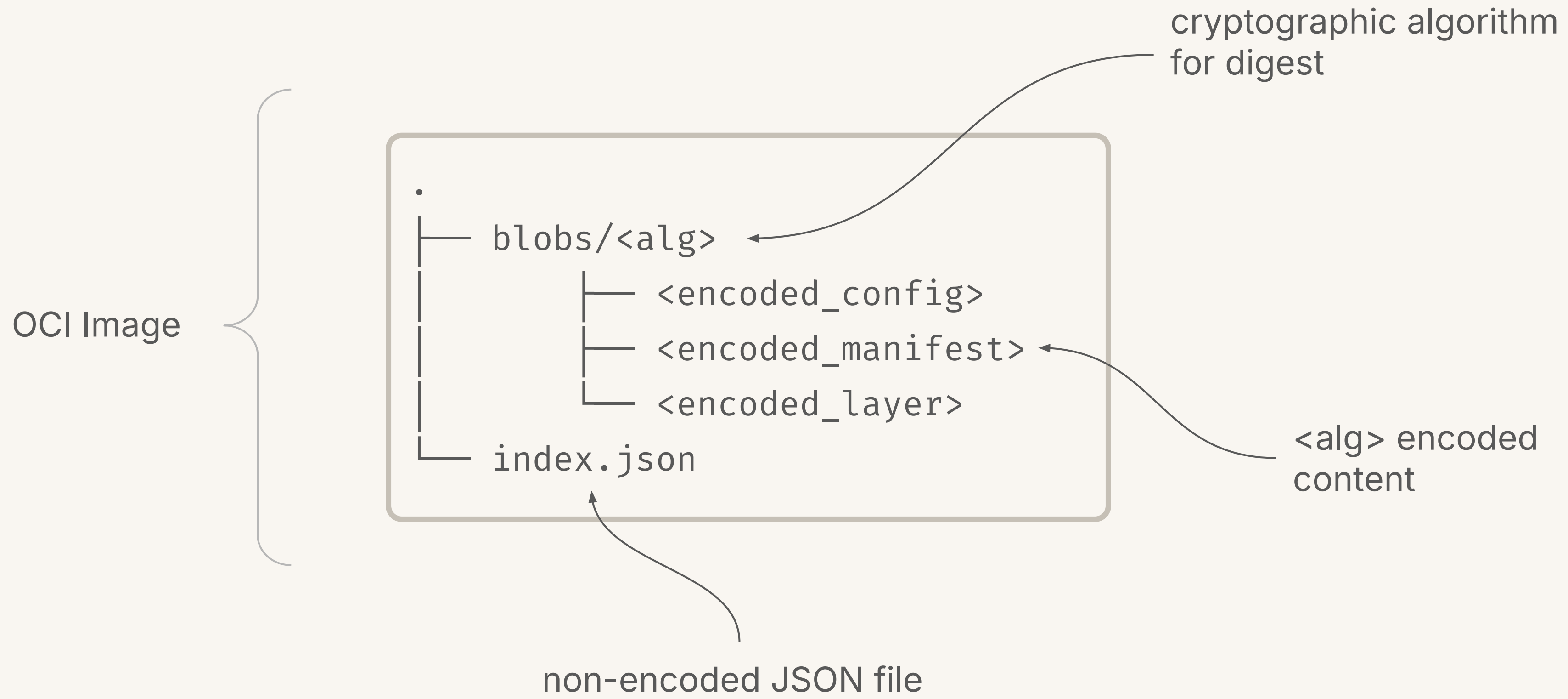


```
FROM scratch

COPY ./hello ./

ENTRYPOINT ["./hello"]
```

```
podman build -f Containerfile .
```

image layout

the OCI image spec layout



```
.
├── blobs/<alg>
│       ├── <encoded_config>
│       ├── <encoded_manifest>
│       └── <encoded_layer>
└── index.json
```

scratch image: layer

first and only layer with a binary

gzipped
tar archive

```
$ gcc -o hello hello.c -static
$ tar -czvf layer.tar.gz hello

$ sha256sum layer.tar.gz
c37c06cdec9d6a0f2a2d5 layer.tar
$ mv layer.tar c37c06cdec9d6a0f2a2d5


$ tree ../..
└── blobs/sha256
    └── c37c06cdec9d6a0f2a2d5
```

content
addressable
archive

layout

scratch image: config

setting the entrypoint and other config options

platform

entrypoint

```
$ cat config.json
{
    "architecture": "amd64",
    "os": "linux",
    "config": {
    "Entrypoint": [
        "./hello"
        ]
    }
}

$ mv config.json $(sha256sum
config.json | awk '{print $1}')
```

scratch image: manifest

identifying the layers and the config

```
$ cat manifest.json
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.oci.image.manifest.v1+json",
  "config": {
     "mediaType": "application/vnd.oci.image.config.v1+json",
     "digest": "sha256:bf7031d43f7d2c9ec77ed",
     "size": 296
  },
  "layers": [
     {
       "mediaType": "application/vnd.oci.image.layer.v1.tar+gzip",
       "digest": "sha256:583eb9106f1be6dffa3f0",
       "size": 1377804
     }
  ]
}

$ mv manifest.json $(sha256sum manifest.json | awk '{print $1}')
```

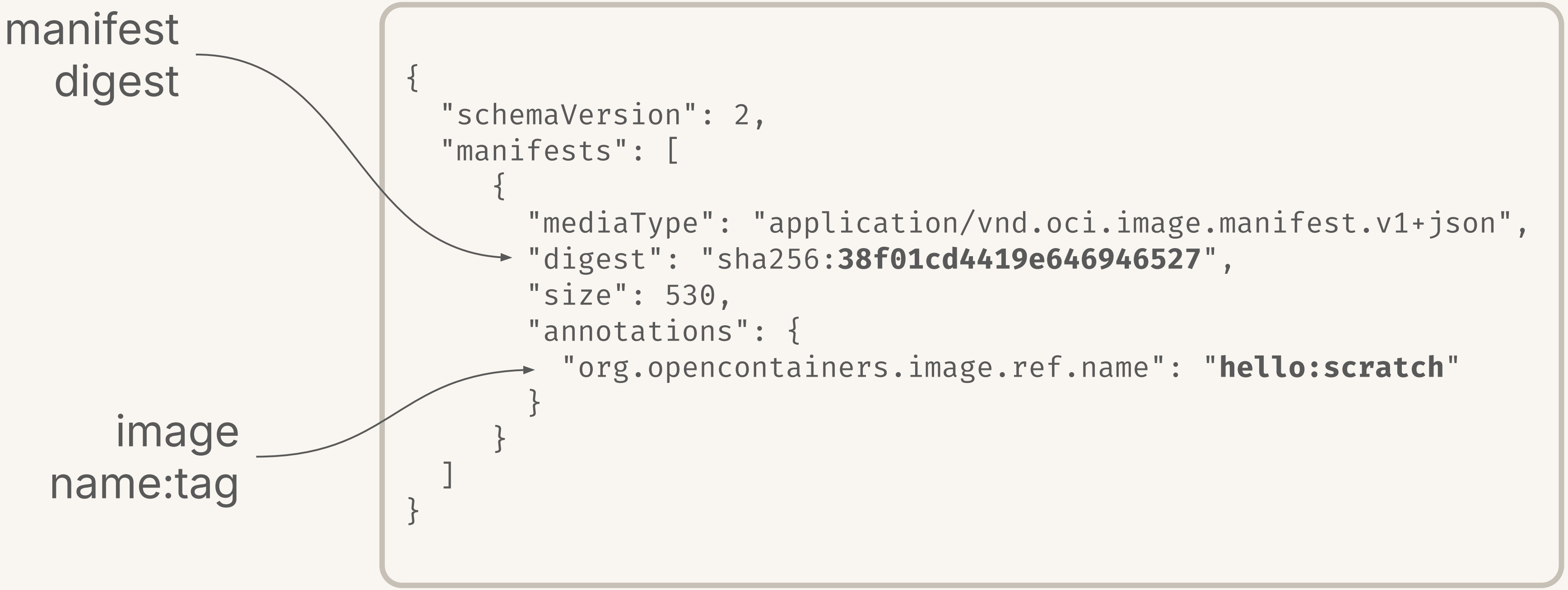
content descriptor

config

layer digest

scratch image: index

identifying the only manifest in our image



```
{
  "schemaVersion": 2,
  "manifests": [
     {
       "mediaType": "application/vnd.oci.image.manifest.v1+json",
       "digest": "sha256:38f01cd4419e646946527",
       "size": 530,
       "annotations": {
         "org.opencontainers.image.ref.name": "hello:scratch"
       }
     }
  ]
}
```

scratch image: load & verify

```
$ tree
├── blobs/sha256
│    ├── 38f01cd4419e646946527
│    ├── c37c06cdec9d6a0f2a2d5
│    └── cd12bca58eb0ed29b24e4
└── index.json

$ tar -cf hello.tar *
```

```
$ podman load < hello.tar
Getting image source signatures
Copying blob c37c06cdec9d done   |
Copying config cd12bca58e done   |
Writing manifest to image destination
Loaded image: localhost/hello:scratch

$ podman image ls hello
REPOSITORY   TAG        IMAGE ID        SIZE
hello        scratch        25e8b3bd9720    3.67MB

$ podman run localhost/hello:scratch world
hello world!
```

demo!

alpine base image

first variant but *with* alpine base image

minimal root filesystem

static binary

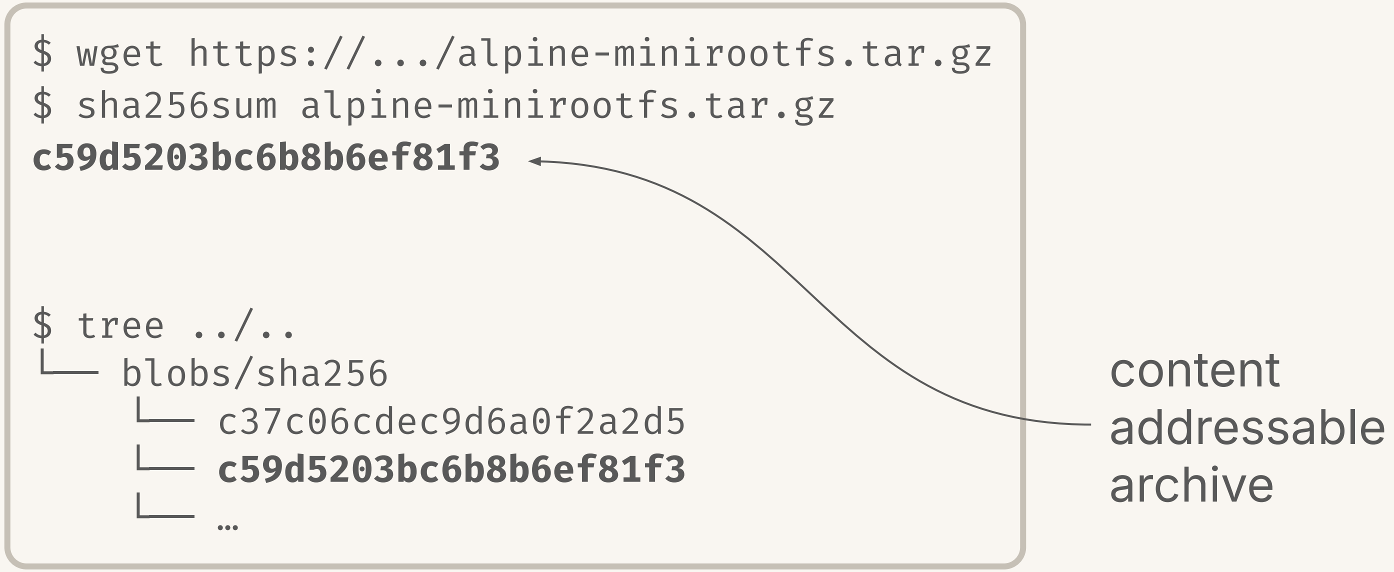
`time`d entrypoint

```
FROM alpine

COPY ./hello ./

ENTRYPOINT [“time”, “./hello”]
```

```
podman build -f Containerfile .
```

alpine base: layer

using an alpine base root filesystem



```
$ wget https://.../alpine-minirootfs.tar.gz
$ sha256sum alpine-minirootfs.tar.gz
c59d5203bc6b8b6ef81f3


$ tree ../..
└── blobs/sha256
    └── c37c06cdec9d6a0f2a2d5
    └── c59d5203bc6b8b6ef81f3
    └── …
```

alpine base: manifest

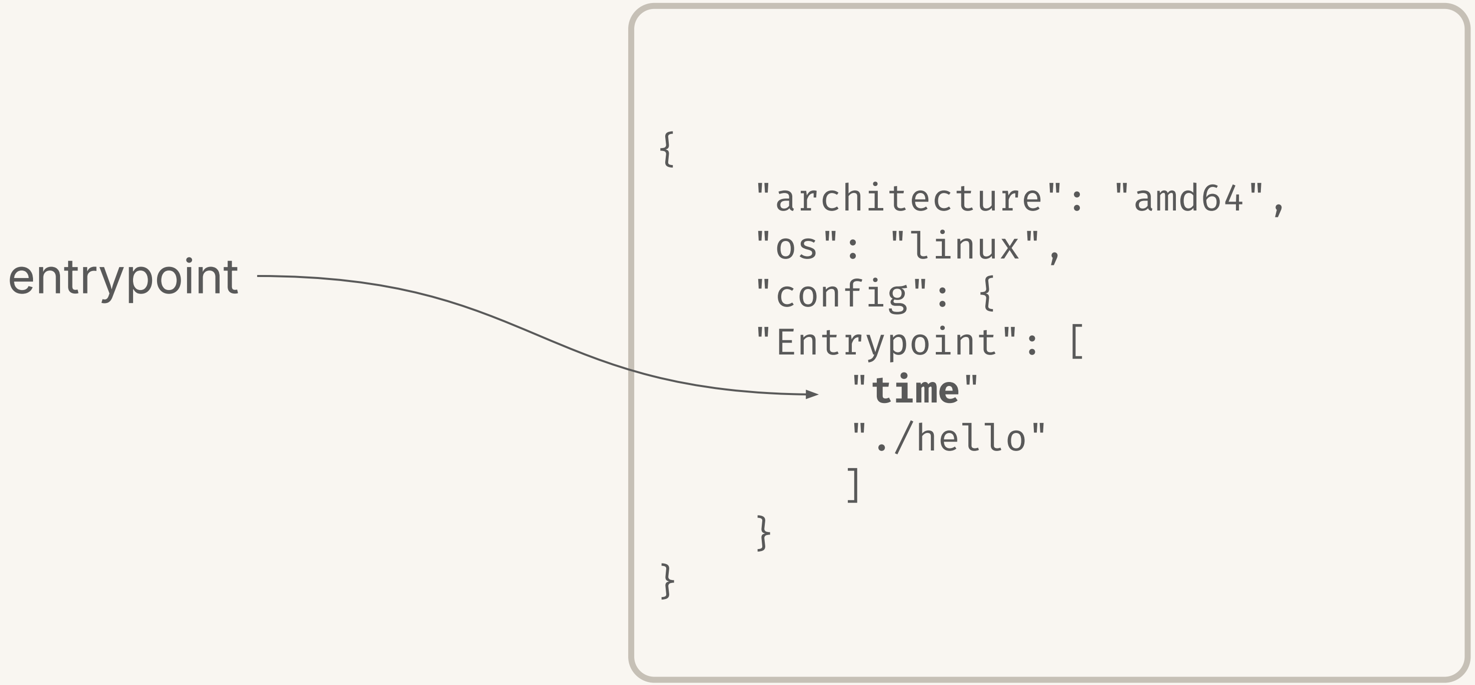
Identifying both the layers and the config

```
"config": {
   "mediaType": "application/vnd.oci.image.config.v1+json",
   "digest": "sha256:bf8a32d43f7d2c9ec89fb",
   "size": 296
},
"layers": [
   {
     "mediaType": "application/vnd.oci.image.layer.v1.tar+gzip",
     "digest": "sha256:c59d5203bc6b8b6ef81f3",
     "size": 3279768
   },
   {
     "mediaType": "application/vnd.oci.image.layer.v1.tar+gzip",
     "digest": "sha256:c37c06cdec9d6a0f2a2d5",
     "size": 1372611
   }
]
```

layer hashes

alpine base: config

modify the entrypoint



```
{
    "architecture": "amd64",
    "os": "linux",
    "config": {
    "Entrypoint": [
        "time"
        "./hello"
        ]
    }
}
```

alpine base: index

updating the manifest digest and image tag

manifest
digest

image
name:tag

```
{
  "schemaVersion": 2,
  "manifests": [
     {
       "mediaType": "application/vnd.oci.image.manifest.v1+json",
       "digest": "sha256:38f01cd4419e646946527",
       "size": 530,
       "annotations": {
         "org.opencontainers.image.ref.name": "hello:alpine"
       }
     }
  ]
}
```

demo!

fin

- → container image *internals*
- → how *layers* are used
- → *multi-platform* images

resources

https://github.com/opencontainers/image-spec

https://danishpraka.sh/posts/build-a-container-image-from-scratch/

https://danishpraka.sh/static/build-a-container-image-from-scratch.pdf

danishpraka.sh
Software Engineer
SUSE