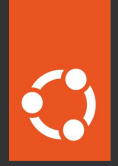
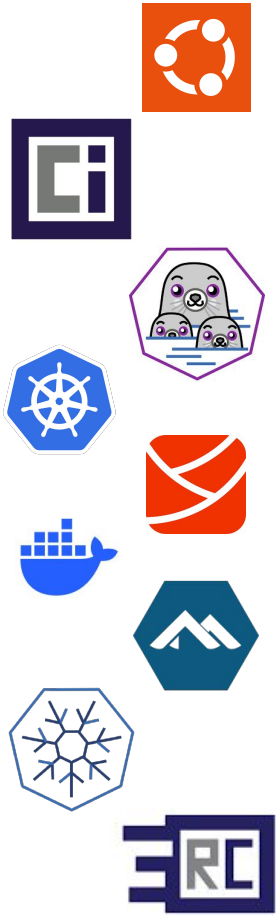


Open Source 101

Recipes & Runtimes
Making Sense of Containers in 2026

Soundarya Rangarajan
Engineering @ Canonical
16th June 2026

How did we get here?



2026

Provenance
Minimal attack surface
Supply-chain poisoning
….

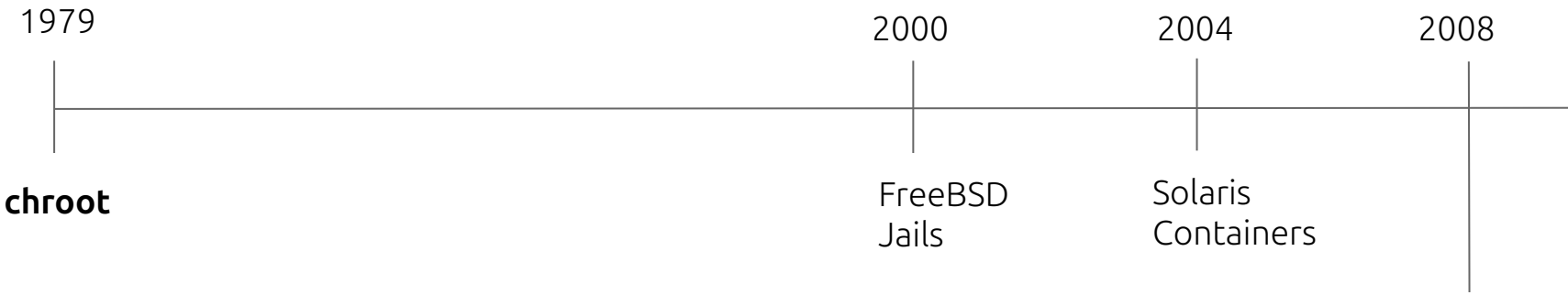
Let’s rewind…

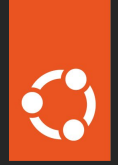
2013

2015

2019

I mean, *really* rewind





Let’s containerize our micro service!

~~Let’s containerize our micro service!~~

Let’s level-up our micro bakery! 

Disclaimer:
I know nothing about baking.
Take my analogies with a pinch of salt (or should I say sugar?)

Baking at home and shipping to customers

Baking in cloud kitchens around the city and shipping to customers nearby

Recreate the same taste. Every single time.

Ensure the whole kitchen doesn’t go down if something goes wrong with our baking.

Recreate the same taste. Every single time.

Reproducible environment.

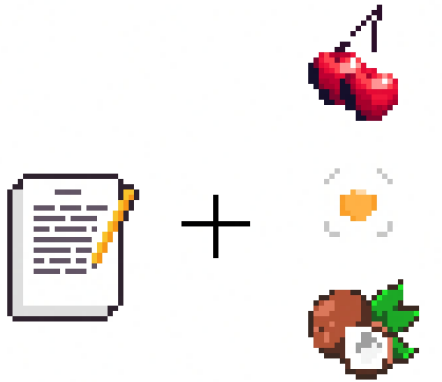
Ensure the whole kitchen doesn’t go down if something goes wrong with our baking.

Isolation.

#1 Container Image

Bake kit =

Recipe + Ingredients



A container image is a blueprint to create a running container.

It contains everything needed to run an app:

Code

Runtime

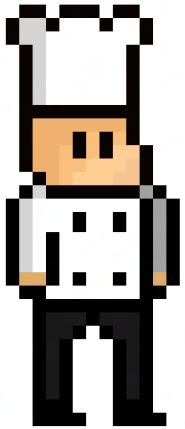
Libraries

Env, settings

#2 Container

A chef,

In an isolated part of a cloud kitchen

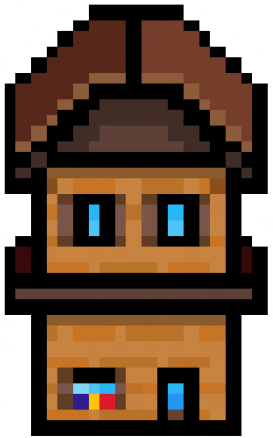


A container is the running process actually serving your app.

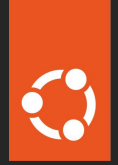
“Isolated, lightweight process”

#3 Container runtime

An isolated part of a cloud kitchen, just for your baking



The specialized software component that takes a container image, unpacks it, and turns it into a running isolated process (a container) on a host OS



Long time, no code

Time for a practical example

`git checkout v0-basic-webapp`
To inspect the code and tinker

#4 Image hardening

Fewer ingredients = tighter control

Attack surface:

Every component that could potentially contain a CVE, or be abused if compromised.

Image hardening

Goal (i) : Reduce **exposure**. What are the chances things could go wrong?

Reduce vulnerability exposure.

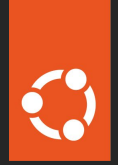
Goal (ii) : Reduce **blast radius**. If things do go wrong, just how bad can it get?

Reduce post-compromise capability.

Image hardening: *How?*

1. Remove unnecessary packages

   Multi-stage builds and `slim` base images.
   Don’t add build deps in final image. Only runtime deps.

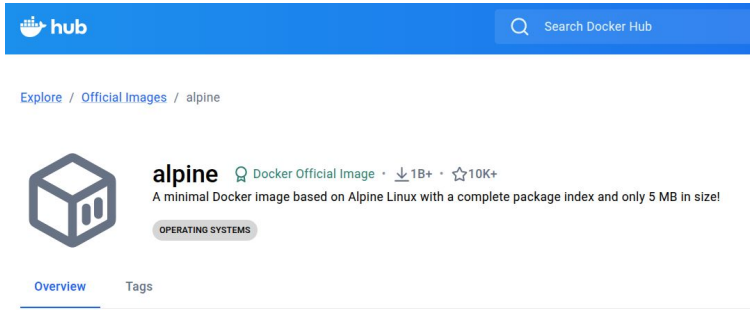


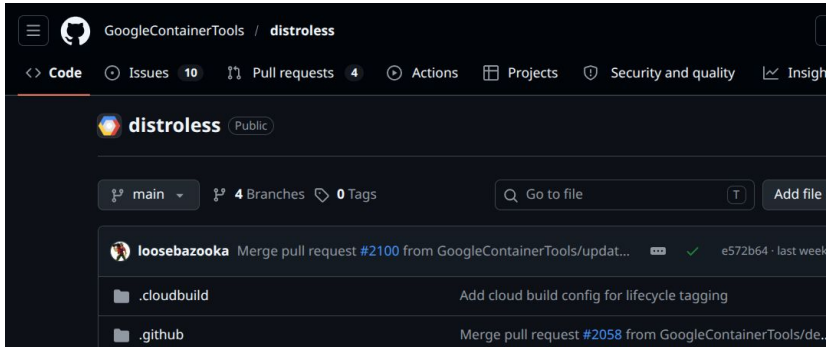
Let’s optimize our naive Dockerfile…

`git checkout v2-multistage`
To inspect the code and tinker

Image hardening: *How?*

2. Minimal base image
    a. Alpine
    b. Distroless
    c. Chisel





Comparing the approaches

IMAGE	ID	DISK USAGE	CONTENT SIZE	EXTRA
dessert-shop:naive	7209228cba2d	1.79GB	414MB	U
dessert-shop:v2	d6f729c5a686	330MB	80.7MB	U
dessert-shop:v3	b84048dba141	208MB	53.4MB	U

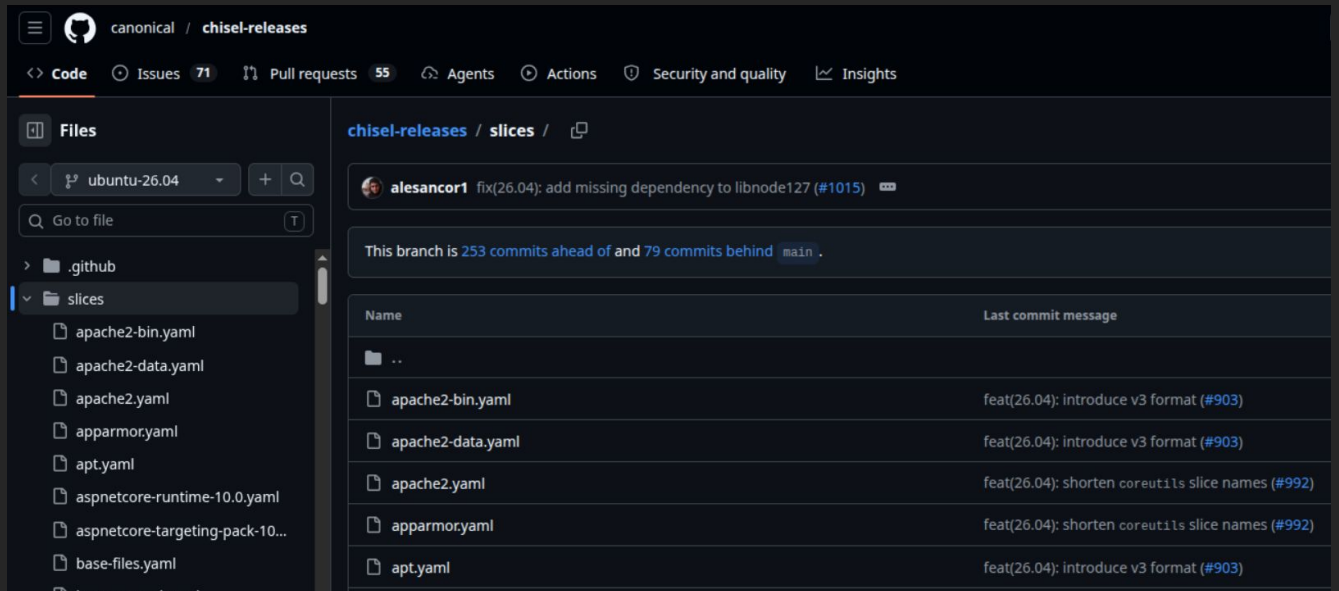
Rockcraft: an open-source tool to build OCI-compliant container images.

- **Declarative**. No imperative scripts.
- **Pebble**. Process orchestrator.
- **Chisel**! Trusted Ubuntu, but *just* what you need.

Let’s try it out…

`git checkout v4-rockcraft`
To inspect the code and tinker

Chisel slices repo



#5 Non-root users

Image hardening isn’t just about a smaller image.
It also involves stricter runtime restrictions.

CVE-2019-5736 : Container escape vulnerability

#6 Rootless runtimes

User
↓
docker CLI
↓
dockerd (running as HOST ROOT)
↓
runc (running as HOST ROOT)
↓
container

#6 Rootless runtimes

podman

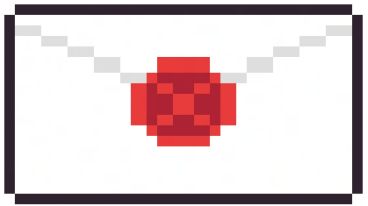
User
↓
docker CLI
↓
dockerd (running as HOST ROOT)
↓
runc (running as HOST ROOT)
↓
container

User (alice)
↓
dockerd-rootless / podman
↓
runc/crun
↓
container

#7 Image signing

Do I know this ingredient is from a trustworthy source?

Sealed: stamp of authority, and a tamper proof



A cryptographic process that verifies the authenticity and integrity of container images

#8 Provenance

Manufacture date, location, batch no., I need to see it all.

How was this container image produced?

Full chain of custody.

#9 SBOMs

A full and complete ingredient list.

Software bill of materials.

Details every operating system package, third-party library, and application dependency inside a container environment.

#10 Attestations

Certification badges on the pack

What facts can be cryptographically asserted about this image?

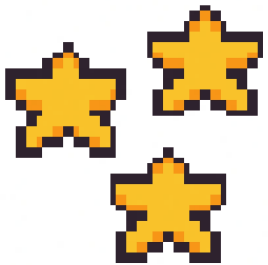
Passed vulnerability scanning

Built from commit abc123

Image has SBOM attached

#11 SLSA

Levels of food safety

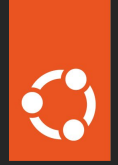


Supply-chain Levels for Software Artifacts

Checklist of standards.

“A common language to talk about how secure software & supply chains are”

https://slsa.dev/



Thank you! Questions?