The Patching Waterfall

Eradicating Container Bloat with Buildpacks

Sai Bharadwaj Avvari


heroku
Cloud computing built and designed for developers.
Joe Kutner
@codefinger


Sai Bharadwaj Avvari

Agenda for this talk

- Chapter 1: The Problem | **The Container Bloat Crisis**
  Drowning in CVE noise and the architectural trap of Hyper-Bloat vs. Fragmentation.

- Chapter 2: The Answer | **Cloud Native Buildpacks and their Anatomy**
  Understanding CNBs and how their fundamental architecture is designed for this use-case.

- Chapter 3: The Scale | **The Patching Waterfall**
  Understanding the tiers and buildpack rebasing at scale.

The Vulnerability Crisis

87%

Vulnerabilities hide in packages **never loaded** at runtime.

Drowning in CVE Noise

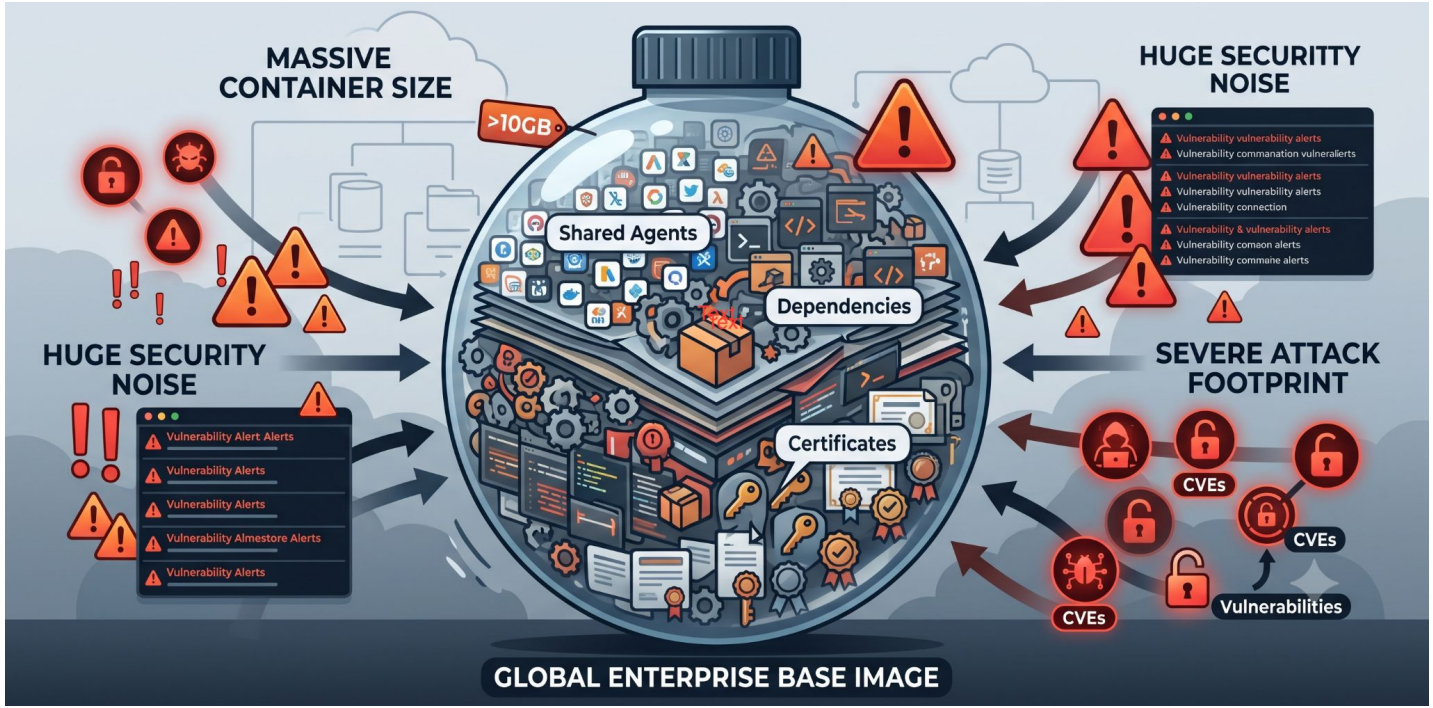
Modern container images are heavily bloated. Platform teams waste thousands of engineering hours triaging vulnerabilities that sit in dormant utilities.

These unused components do more than waste developer focus—they introduce real risk. Attackers actively exploit pre-installed binaries via **Living off the Land (LotL)** patterns.

The Middle-Tier Trap

Option A: Hyper-Bloat

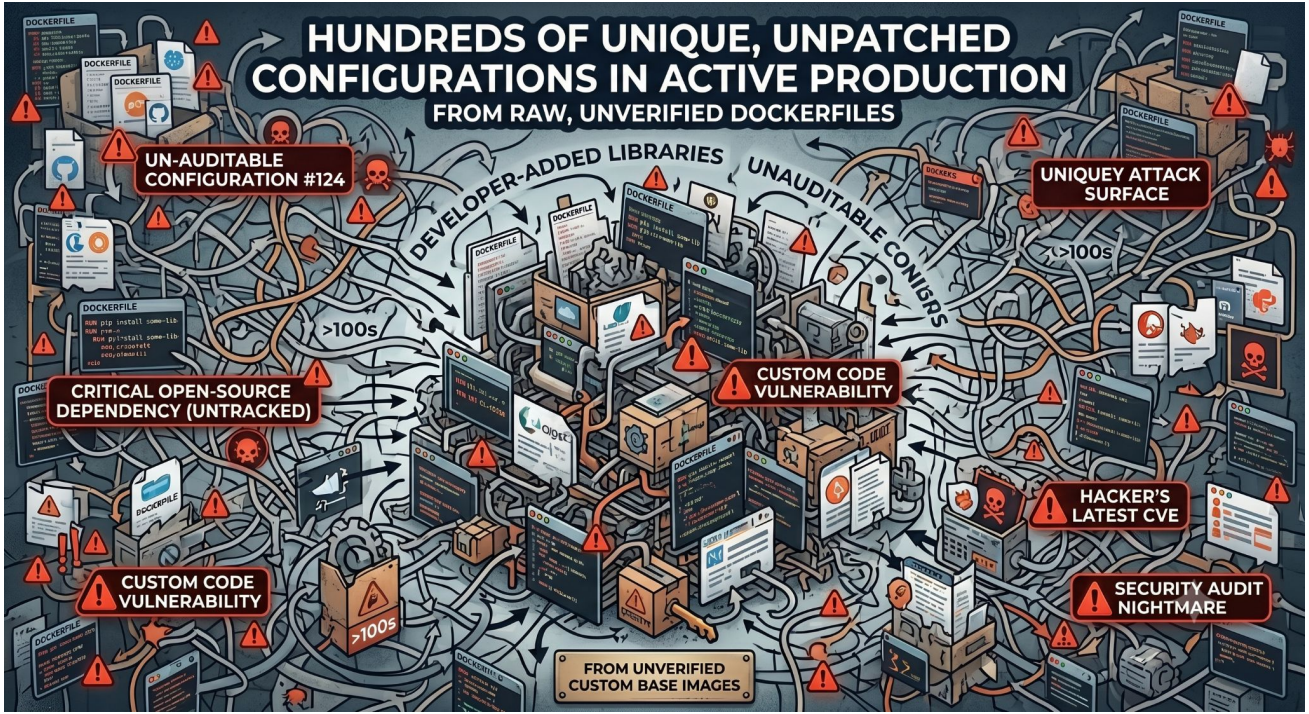
Platforms bake every shared agent, dependency, and certificate into one global enterprise base image.



The Middle-Tier Trap

Option B: Fragmentation

Developers are given custom base images but add their own libraries inside raw, unverified Dockerfiles.




WHAT TO DO? WHAT TO DO?
makeameme.org

Buildpacks!!

What are Buildpacks?

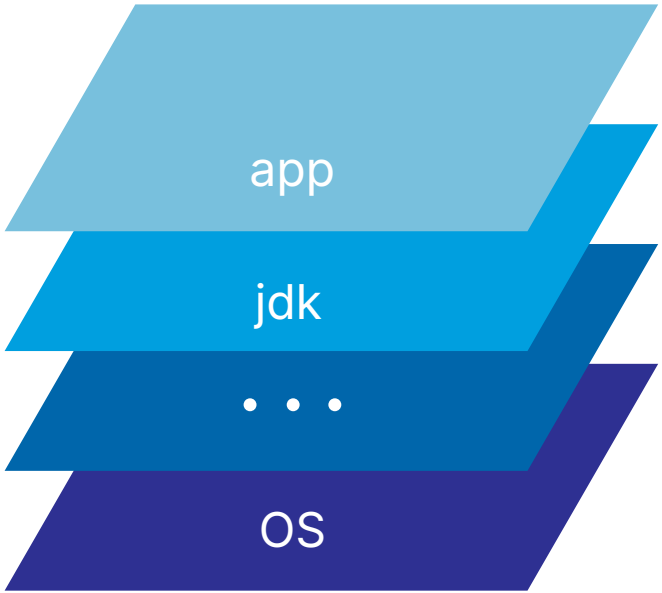
A standard for **converting source code into OCI Images** *without Dockerfiles.*

source

Application Image

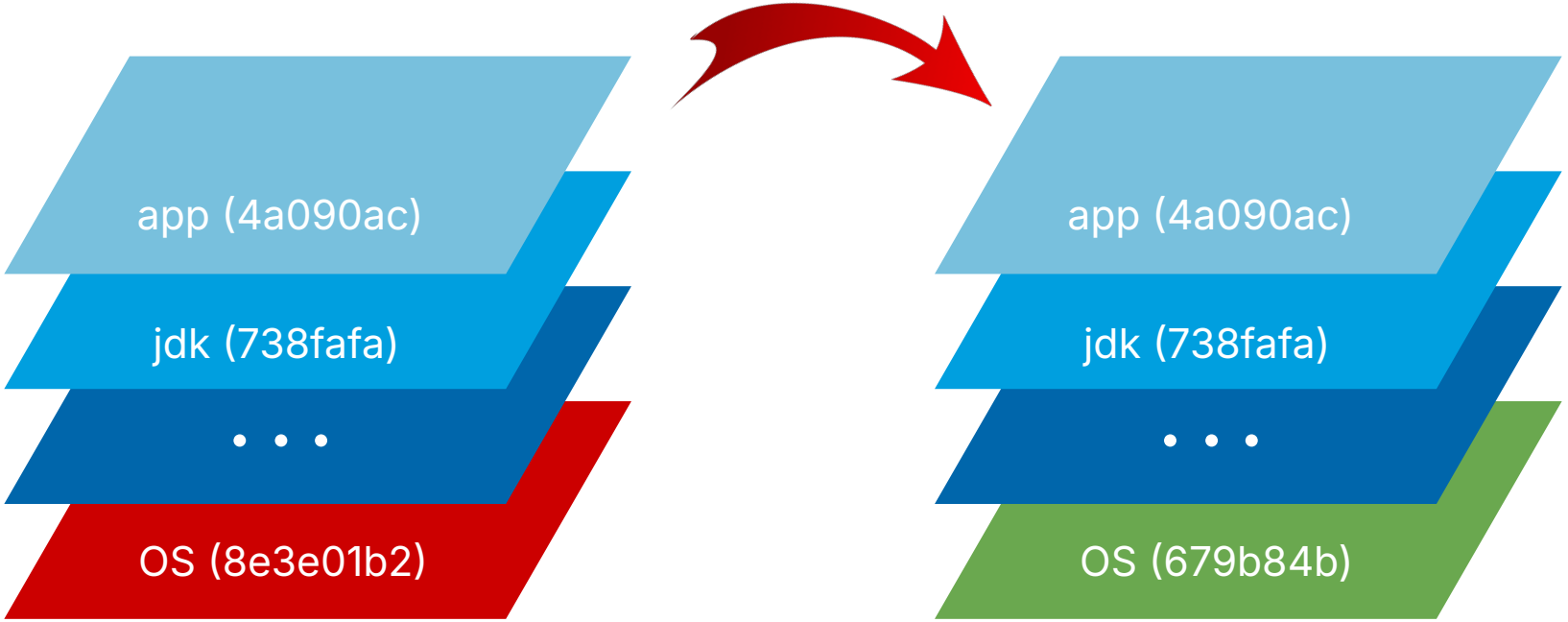
Why use Buildpacks?

Reduce, Reuse, Rebase



Why use Buildpacks?

Reduce, Reuse, Rebase



Who all support Buildpacks?

Bloomberg

VMware Tanzu

DigitalOcean

Google Cloud

The Patching Waterfall

Base OS Tier

Platform SecOps manages minimal run stacks

Middleware Builder

Language runtimes & certificates cached once

App Compilation

Business logic compiled independently of OS

Waterfalls Flows

Security patches flow down without rebuilds

The Patching Waterfall: Architecture

TIER 1: THE BASE STACK

CNB decouples the environment where you **compile** from the environment where you **run**.

```
[stack]
id = "io.buildpacks.stacks.jammy"
build-image = "corp/build-jammy"
run-image = "corp/run-jammy-tiny"
```

TIER 2: CUSTOM BUILDERS

Platform teams compile and cache **domain-specific dependencies** exactly once.

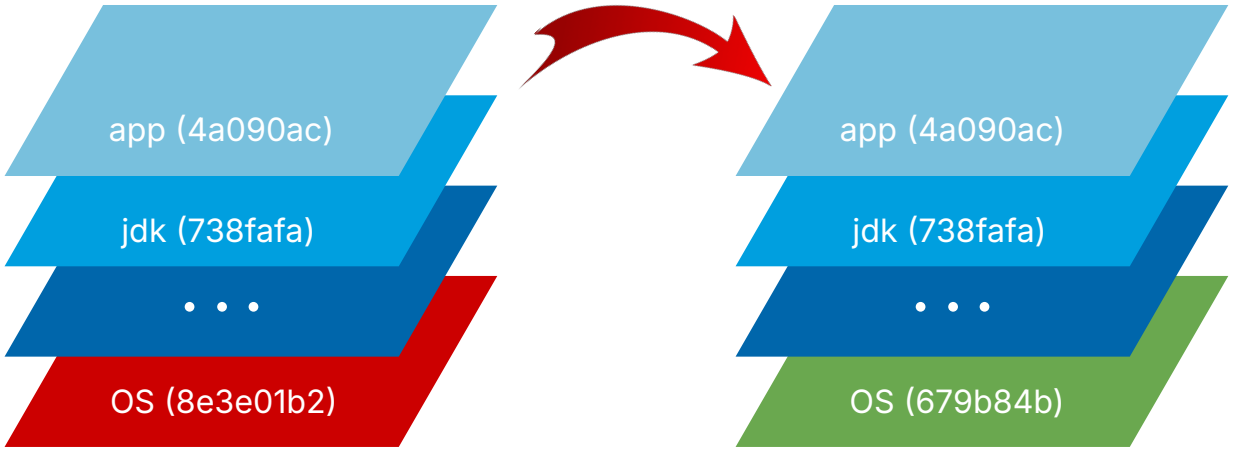
```
# builder.toml
[[buildpacks]]
uri = "docker://gcr.io/paketo-buildpacks/java"

[[order]]
group = [{ id = "corp/ca-certs" }, { id = "corp/java-runtime" }]
```

Rebase: The SHA256 swap


```
# For a single image:
pack rebase my-app:latest

# For the entire fleet (The Waterfall):
kpack-controller --monitor corp/run-base:latest
```

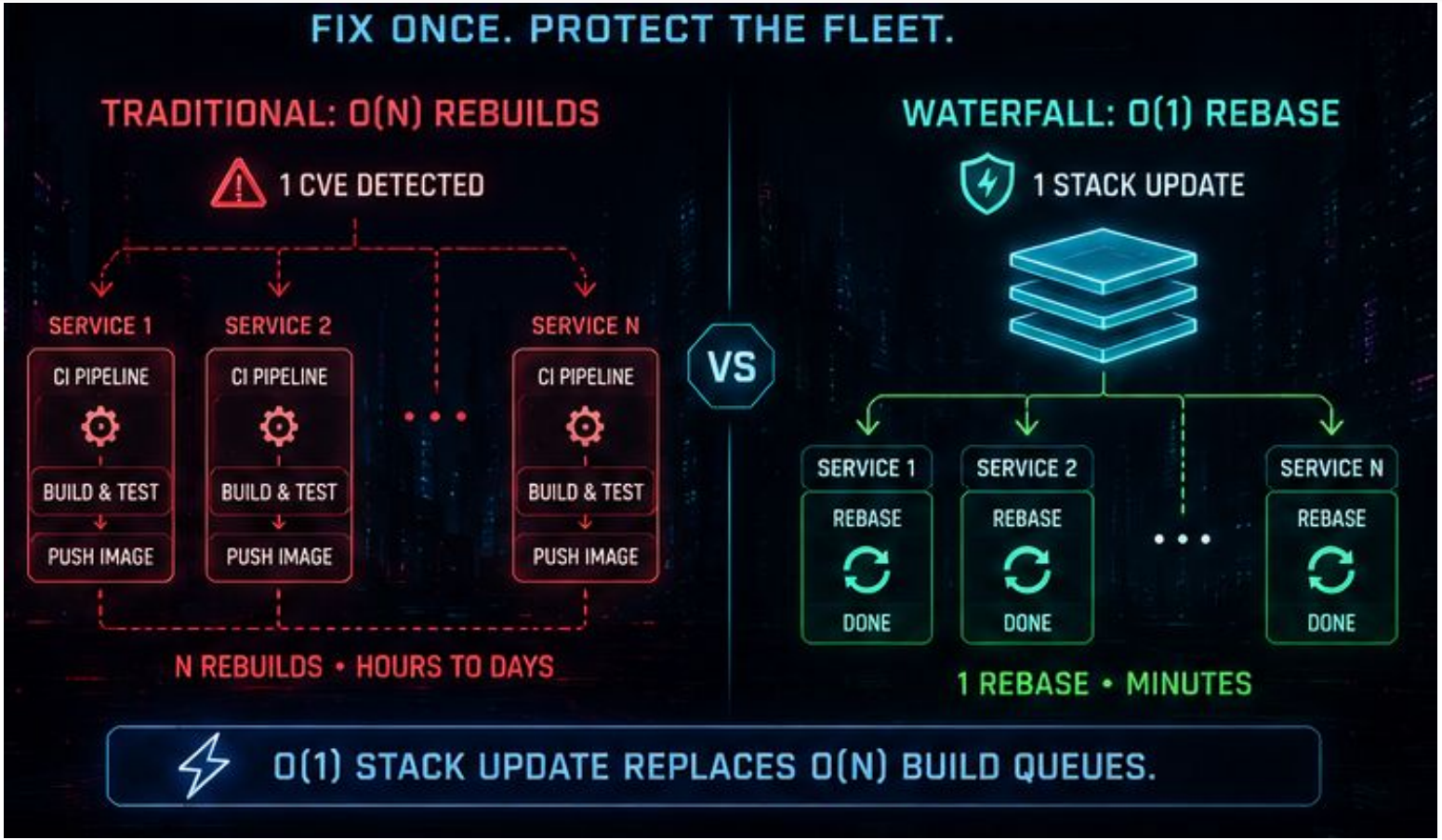


What used to take days will now take minutes with Buildpacks

The Advantages

- O(1) Patch Propagation vs. O(N) Build Queues
- True Separation of Ownership
- Elimination of "Hyper Bloat" and the Attack Surface
- Fleet-wide Consistency and Auditability

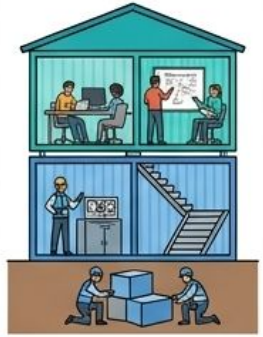
O(1) Patch Propagation v/s O(N) Rebuilds



True Separation of Ownership

FRAGMENTED DOCKERFILES

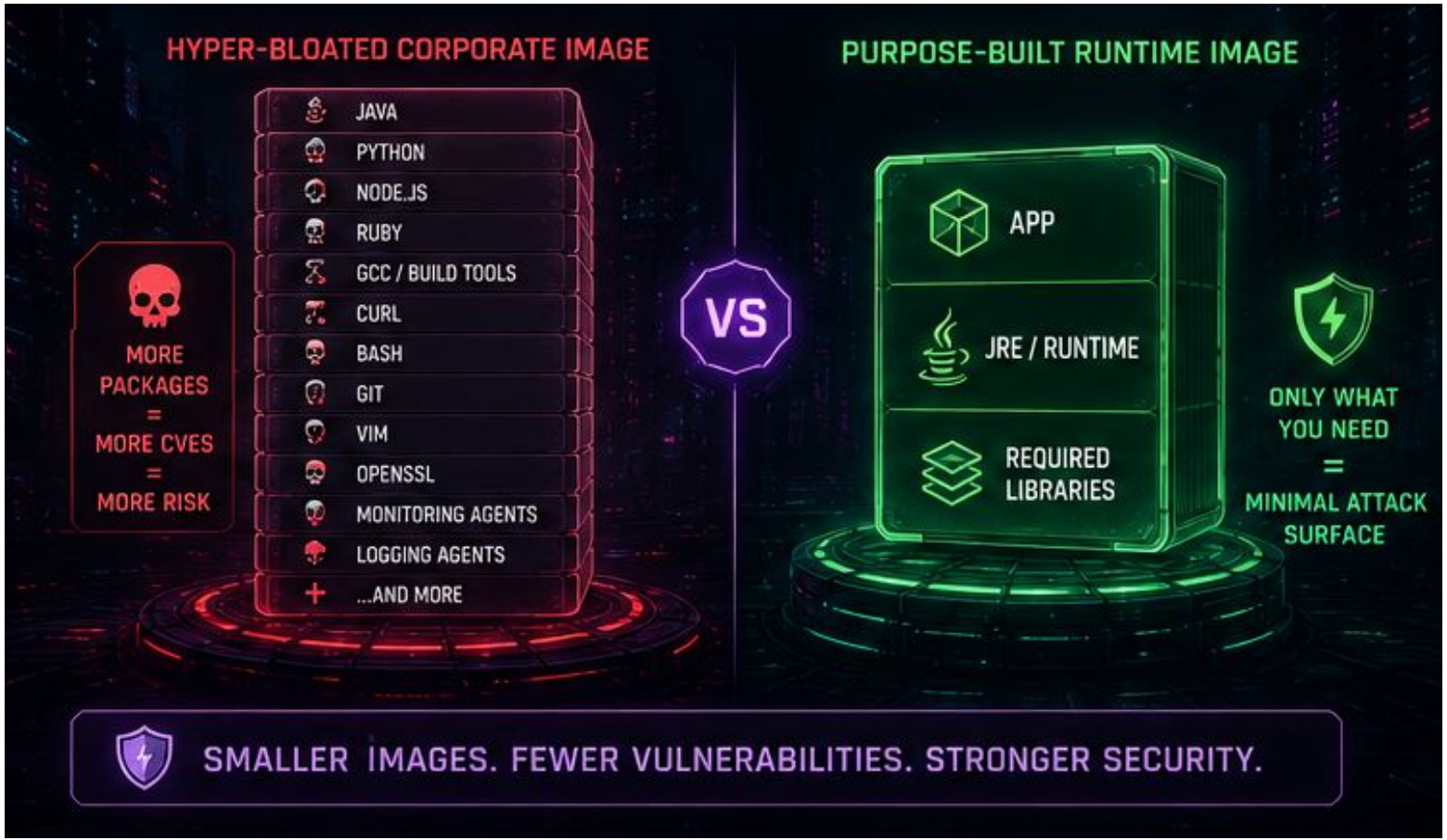
CNB WATERFALL



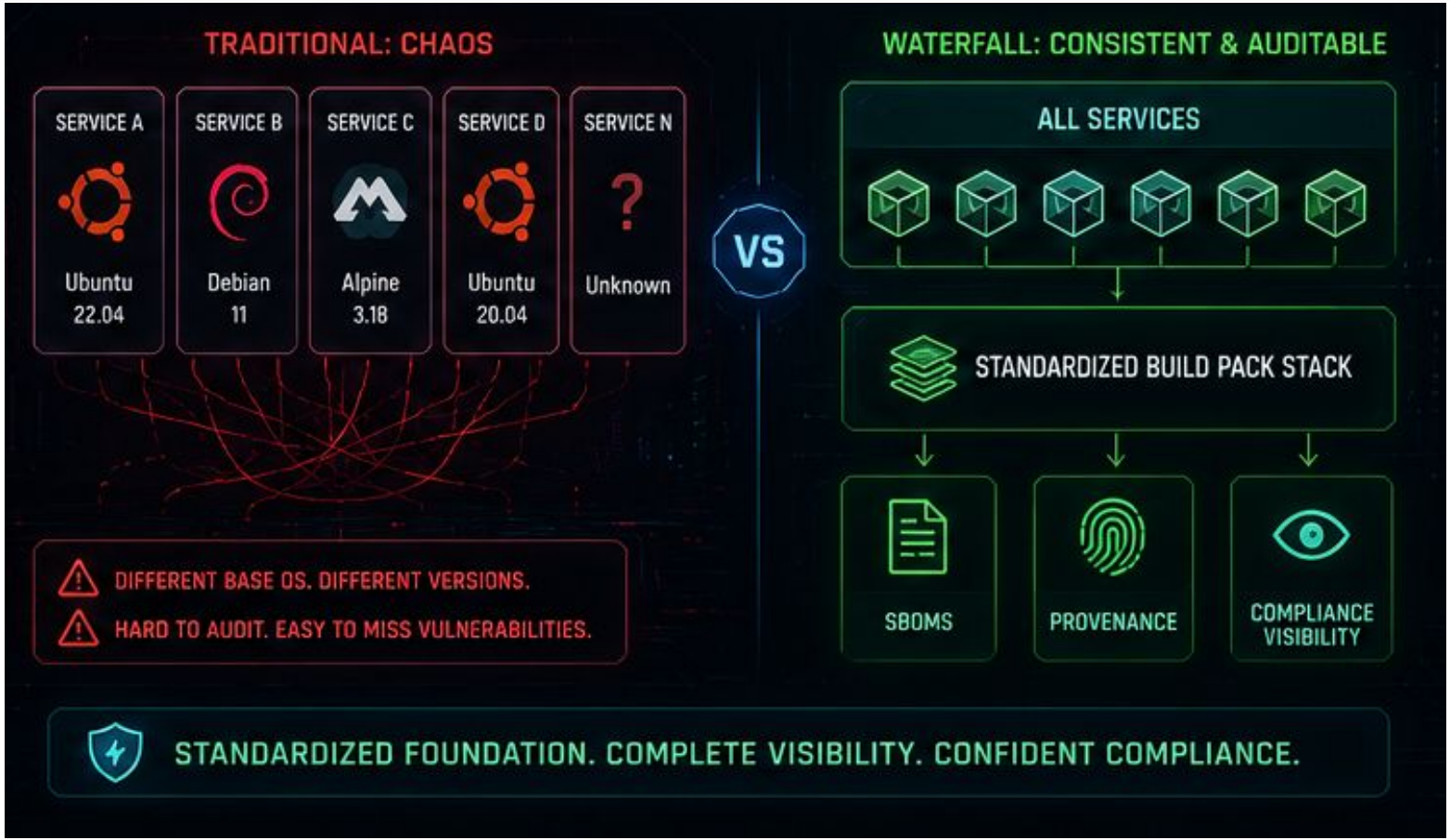
Everyone owns everything
= Nothing is secured

Developers own App Code & User Features.
Security owns Base OS & Compliance.
Updates flow automatically without developer intervention.

Reduction of “Bloat”, i.e., Attack Surface



Fleet-Wide Consistency and Auditability



Learn more about Buildpacks

Let's stop rebuilding and start rebasing.

The Pack CLI – Try it out

```
$ brew tap buildpack/tap
$ brew install pack
$ pack build myimage
```

Engage – Join the Community

- buildpacks.io
- #buildpacks (CNCF Slack)
- github.com/buildpacks

True Separation of Ownership

The App Layer Decoupling

The App Layer

Pure Binaries - No OS Dependencies, no shells, no shared libraries outside the stack contract

The Metadata

CNB stores the SHA256 of the base image