A FIELD REPORT · MELBOURNE RHUG · OCTOBER 2026

From the homelab to your lab

Fedora bootc images, built and boot-tested in GitLab CI,
so servers update like a phone.

Andrew Dunn

andrew@dunn.dev · andrew.dunn.dev

cairn.dunn.dev · gitlab.com/dunn.dev/cairn

andunn@gitlab.com · gitlab.com/andunn · gitlab.com/gitlab-com/public-sector



02 · ABOUT ME

Andrew G. Dunn

WHERE

Grand Rapids, Michigan: my wife, our two sons, cargo bikes.
Family in mid Florida, so Kennedy has been a regular visit.

BEFORE

Modeling and simulation for the Department of Defense, working with several nation states and government agencies, and a brief stint in the space community: NASA, on terramechanics.

NOW

GitLab, a globally distributed open source organization.
I work with regulated industries and build tooling.

THE THREAD

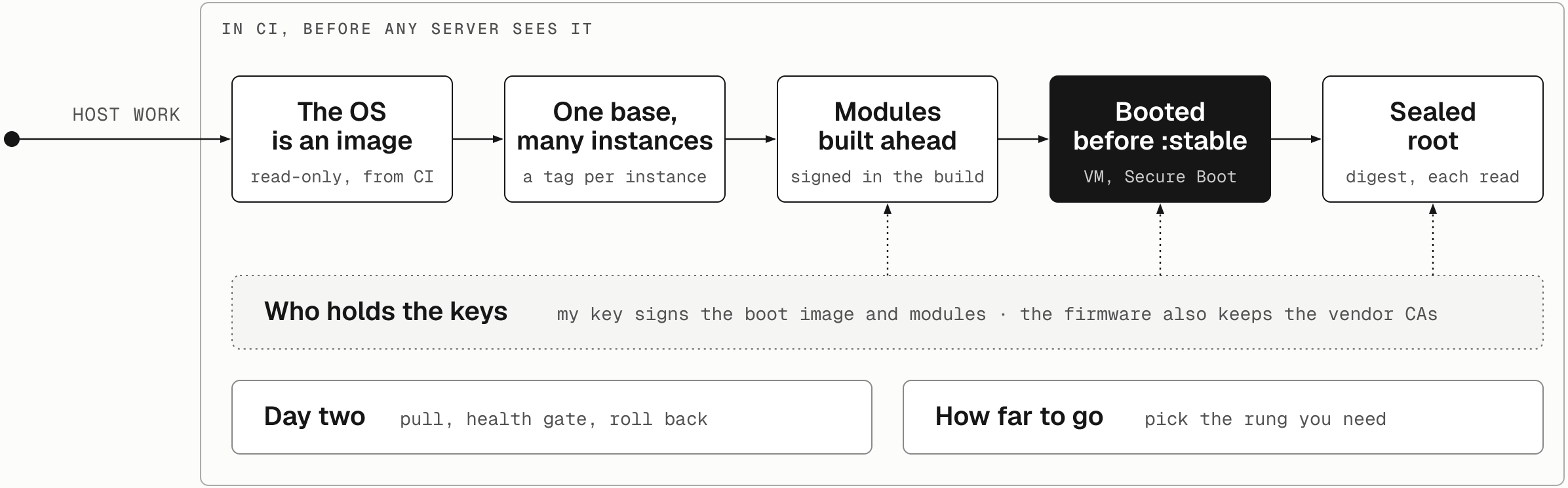
Open source and Linux, from bare metal to large systems, mobile and compact data centers included. I stretch the basic stack a long way before admitting a cluster is needed.
I was an early Podman enthusiast. I never turn off SELinux.



The homelab is where I run that stack, in the open.

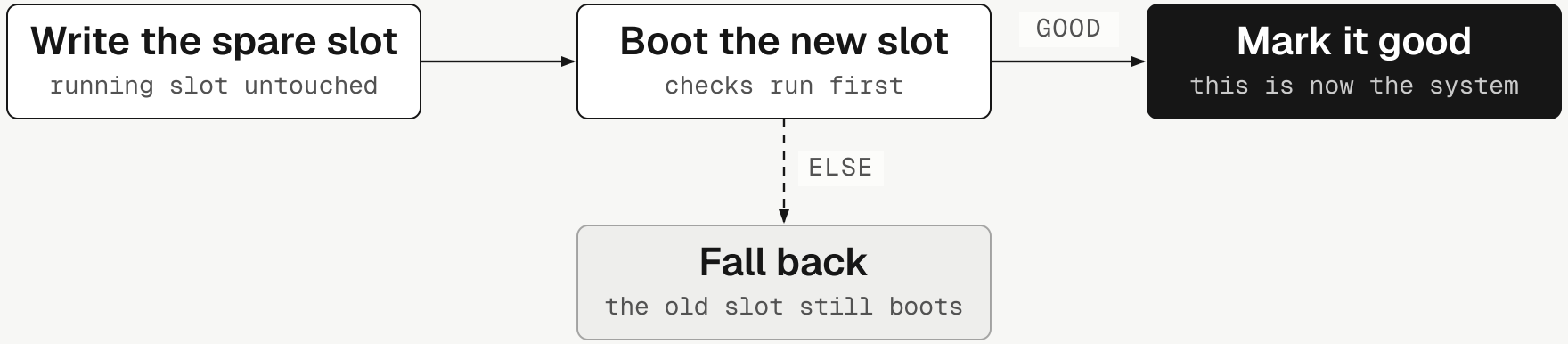
more at andrew.dunn.dev/about

Where this talk is going

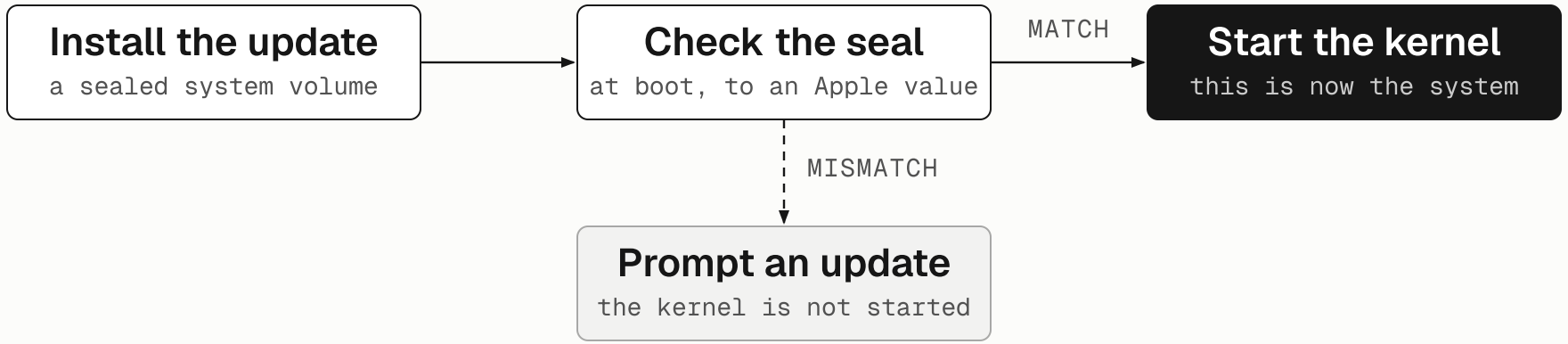


How your phone updates

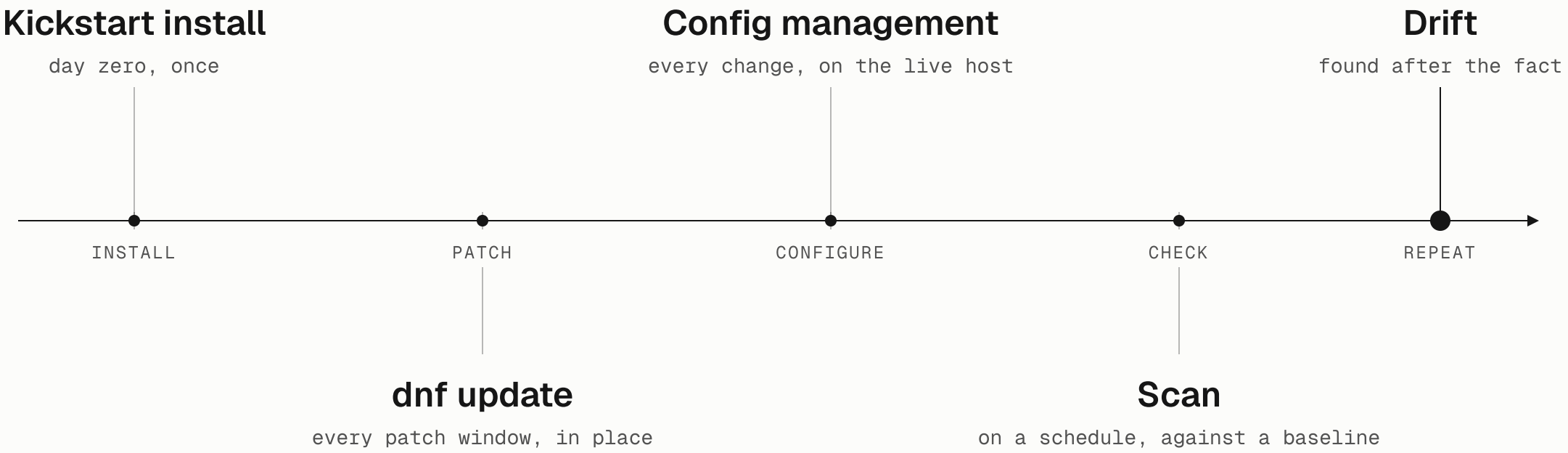
ANDROID



IPHONE



How my servers updated



What the phone had that my servers didn't

PROPERTY	Your phone Android A/B, iPhone sealed volume	My servers, before kickstart, dnf, scans	Image mode, now bootc image, built in CI
Tested before it ships a whole image, as one thing	**Yes** marked good after checks	No patched on the live host	**Yes** boots in CI before the tag moves
Atomic all of it or none of it	**Yes** written to the spare slot	No package by package	**Yes** transactional, in place
Rolls back the last one is kept	**Yes** old slot still boots	No restore from backup	**Opt-in** health gate, opt-in
Verified what runs is what was signed	**Yes** seal checked before the kernel	After the fact a scan against a baseline	**On every read** a digest in the signed boot image

The trade: host work becomes CI work

BEFORE · THE HOST DOES THE WORK

RUNNING HOST

The build list

packages · kernel args · users
firewall zones · units · drivers
applied to the live machine

dnf update –

every patch window, in place

Scan –

against a baseline, after the fact

CHANGES

MOVED
the build list, from the host into the image build

REMOVED
in-place patching on the host

REMOVED
the scan against a baseline

ADDED
a daily schedule that rebuilds only when its resolved inputs changed

CHANGED
drift: a failed build, not a finding on a host

ADDED
pull and reboot, on the host's own timeline; no CI reaches a running host

AFTER · THE BUILD DOES THE WORK

CI · THE IMAGE BUILD

The build list

packages · kernel args · users
zones · units · drivers
in one Containerfile

Daily schedule +

only if inputs changed

IMAGE

RUNNING HOST

Pull and reboot +

on the host's own timeline

A container image is the OS

BUILD

01	**Recipe** a Containerfile anyone can read	mine: Fedora packages into a from-scratch root
02	**Image** an OCI container image, named by its digest, in a registry	same bytes anywhere
03	**bootc** transactional, in-place OS updates from that image	one tool, Fedora or RHEL
04	**Host** boots the image; updates by pulling the next one and rebooting	keeps the last one

RUN

SAME MODEL

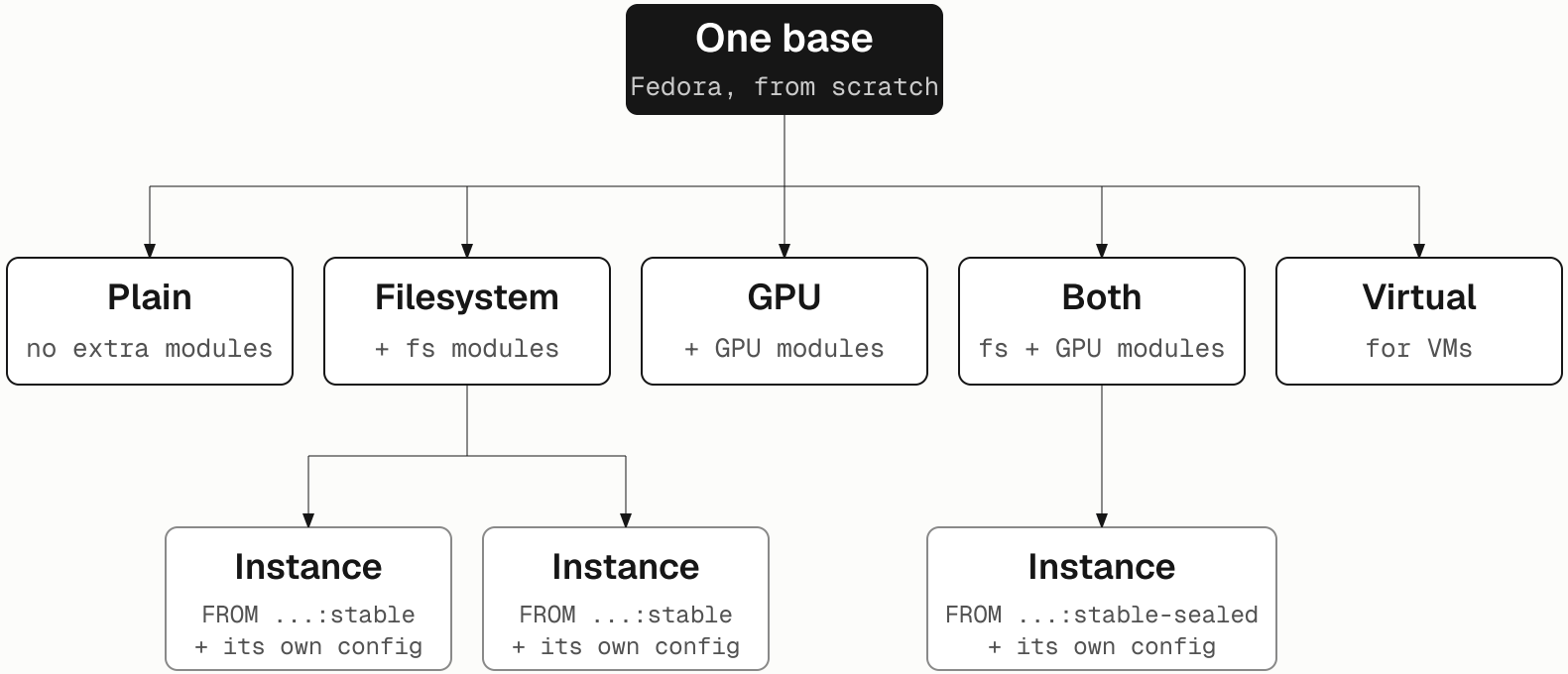
Image mode for RHEL

manages the OS as an OCI image, with bootc, from a Red Hat base.

`rhel10/rhel-bootc`

The difference you see from outside is the FROM line.

Server types and instances; the tag each one follows



BASE

VARIANTS · WHAT THE HARDWARE NEEDS

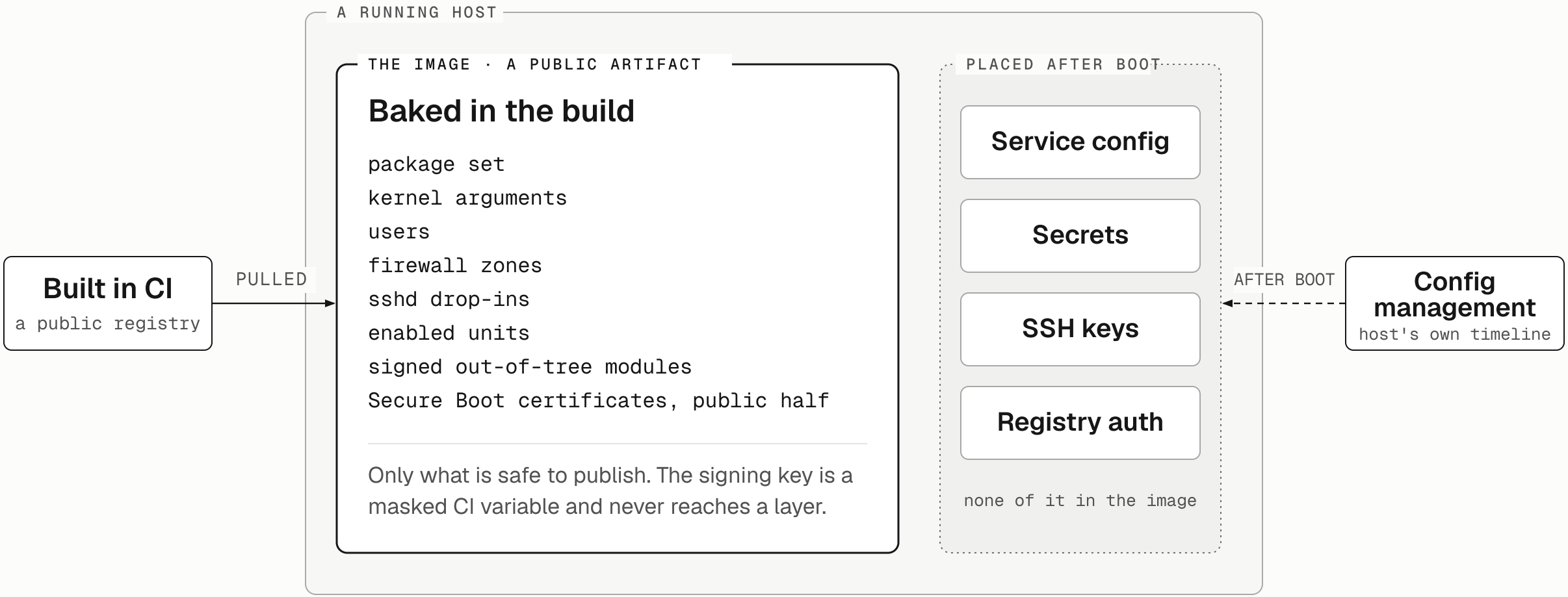
INSTANCES · A TINY REPO EACH

THE TAG THE INSTANCE'S BASE LINE NAMES

`:stable`	the tested digest; :latest moves with it
`:stable-sealed`	the signed, sealed image that booted under Secure Boot

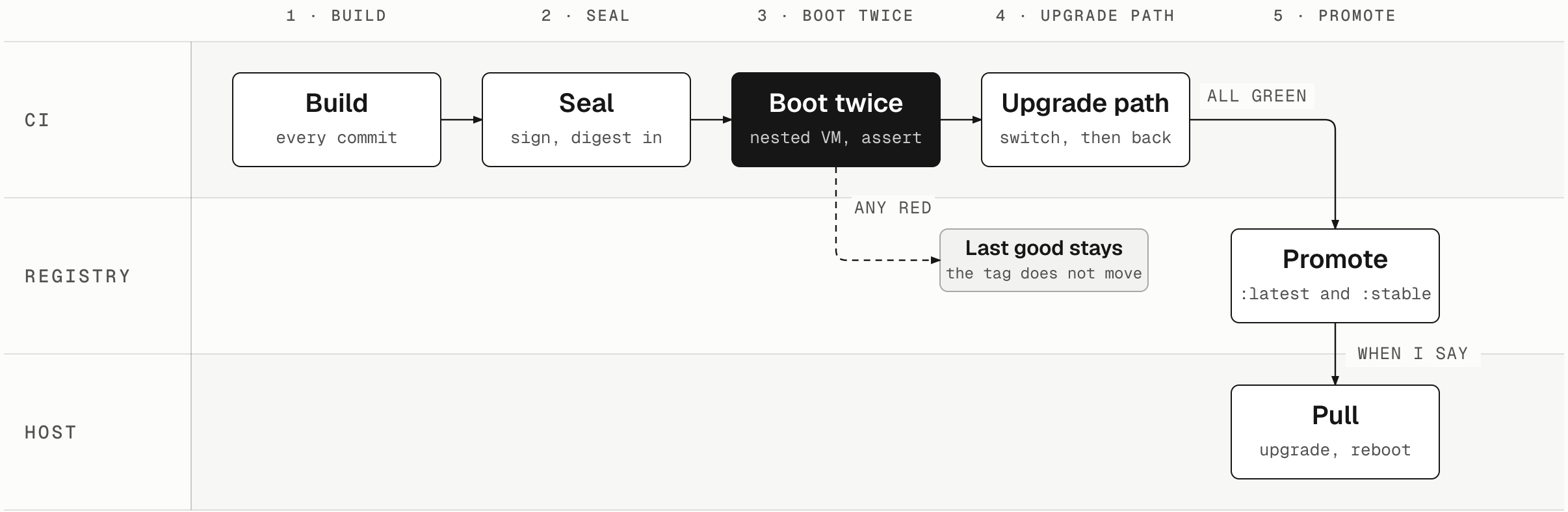
bootc can follow any tag; my hosts follow :stable or :stable-sealed.

What I bake, what arrives at runtime



The build makes a machine with no secrets. After boot it is given its identity.

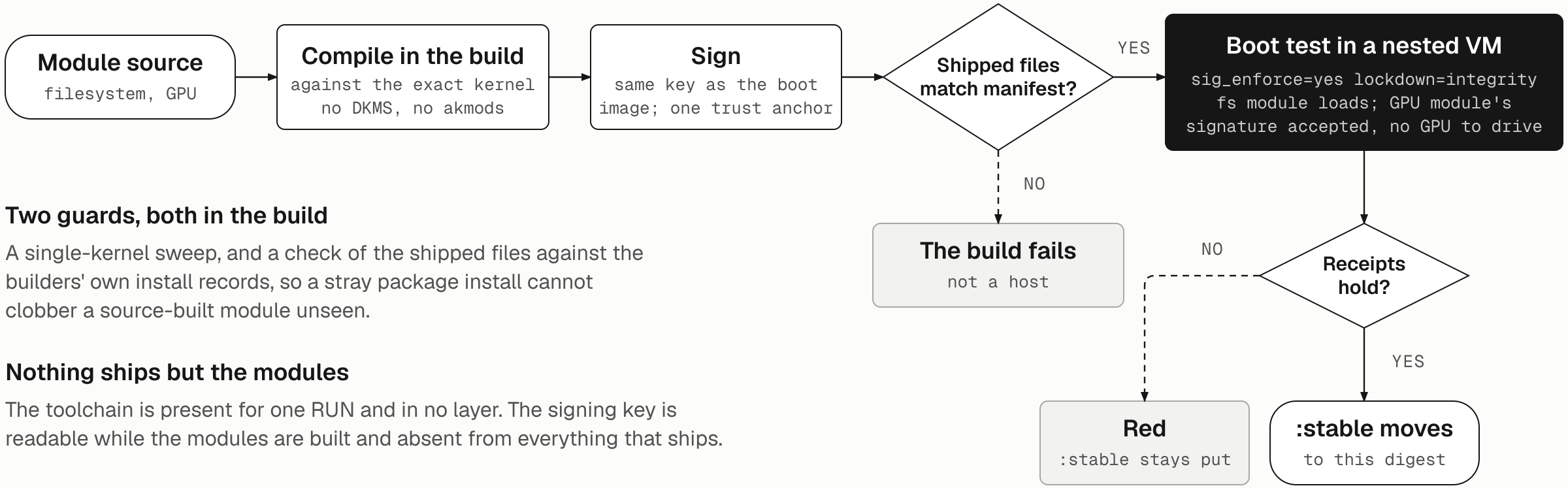
The loop: build, boot-test, promote, pull



No CI reaches a running host; a promotion only posts a trigger.

Second boot: reboot in place, assert again, fail on anything the first shutdown logged at error.
Upgrade path: boot the published image, switch to the candidate, reboot, assert, roll back.

Kernel modules built ahead: built, signed and debugged in CI



Two guards, both in the build

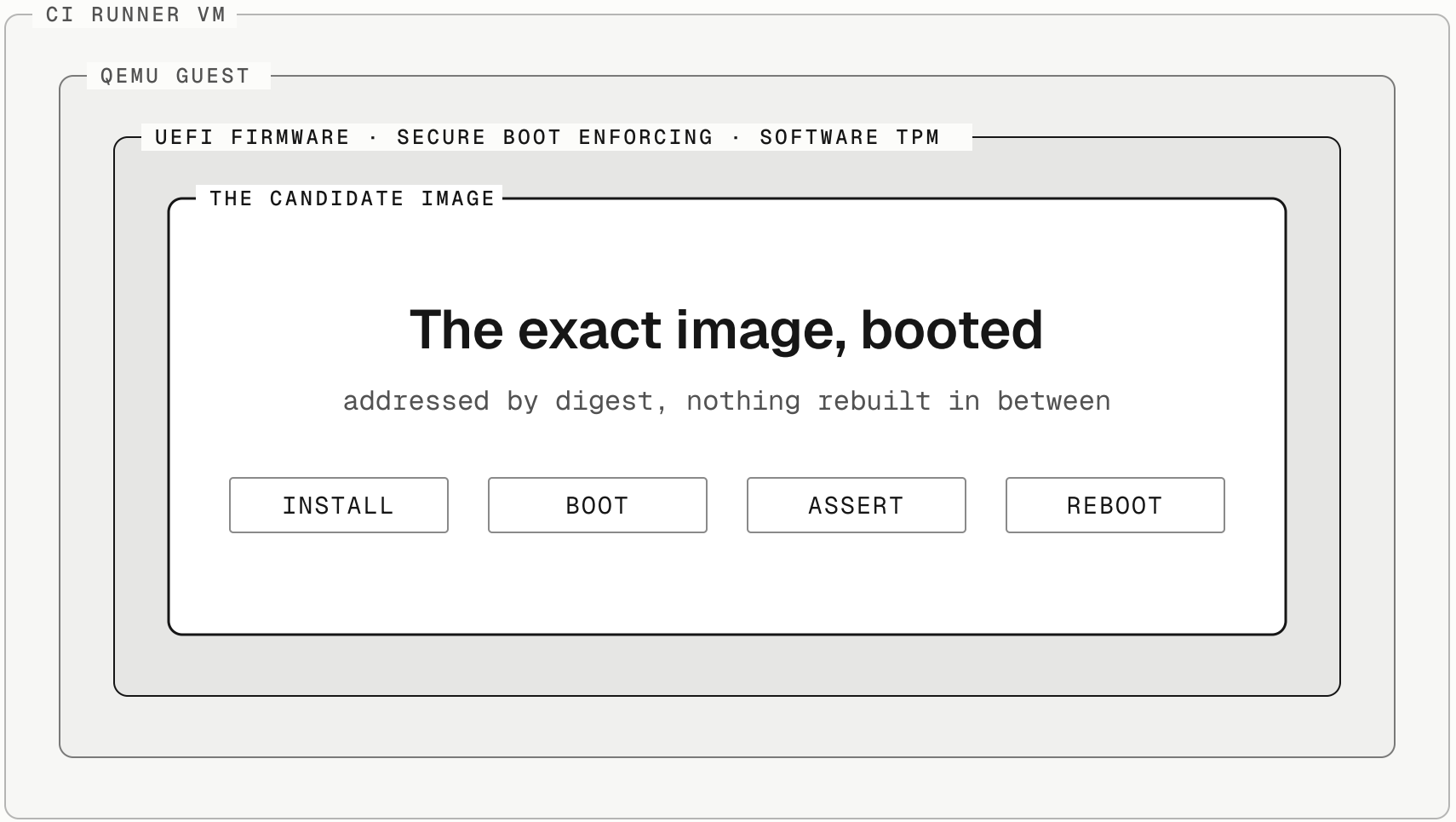
A single-kernel sweep, and a check of the shipped files against the builders' own install records, so a stray package install cannot clobber a source-built module unseen.

Nothing ships but the modules

The toolchain is present for one RUN and in no layer. The signing key is readable while the modules are built and absent from everything that ships.

The negative control: with the signing key unset, the guard must fail closed. It once did not; that job is public and red.

Nested virtualization on a runner



WHAT THE GATE ASSERTS

`secureboot=yes`
`lockdown=integrity`
`sig_enforce=yes`

the root digest on the signed command line

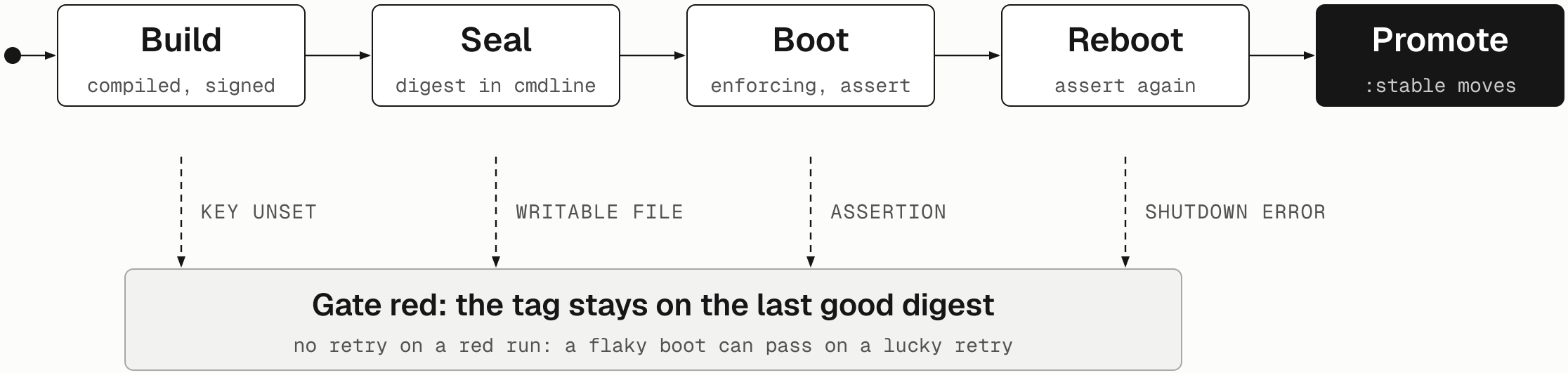
asserted again after an in-place reboot

WHERE IT IS WRITTEN UP

gitlab.com/dunn.dev/cairn/pipeline templates/sealed-instance/README.md

Green and red runs of this gate are public in the base project's CI.

A gate that never fails proves nothing

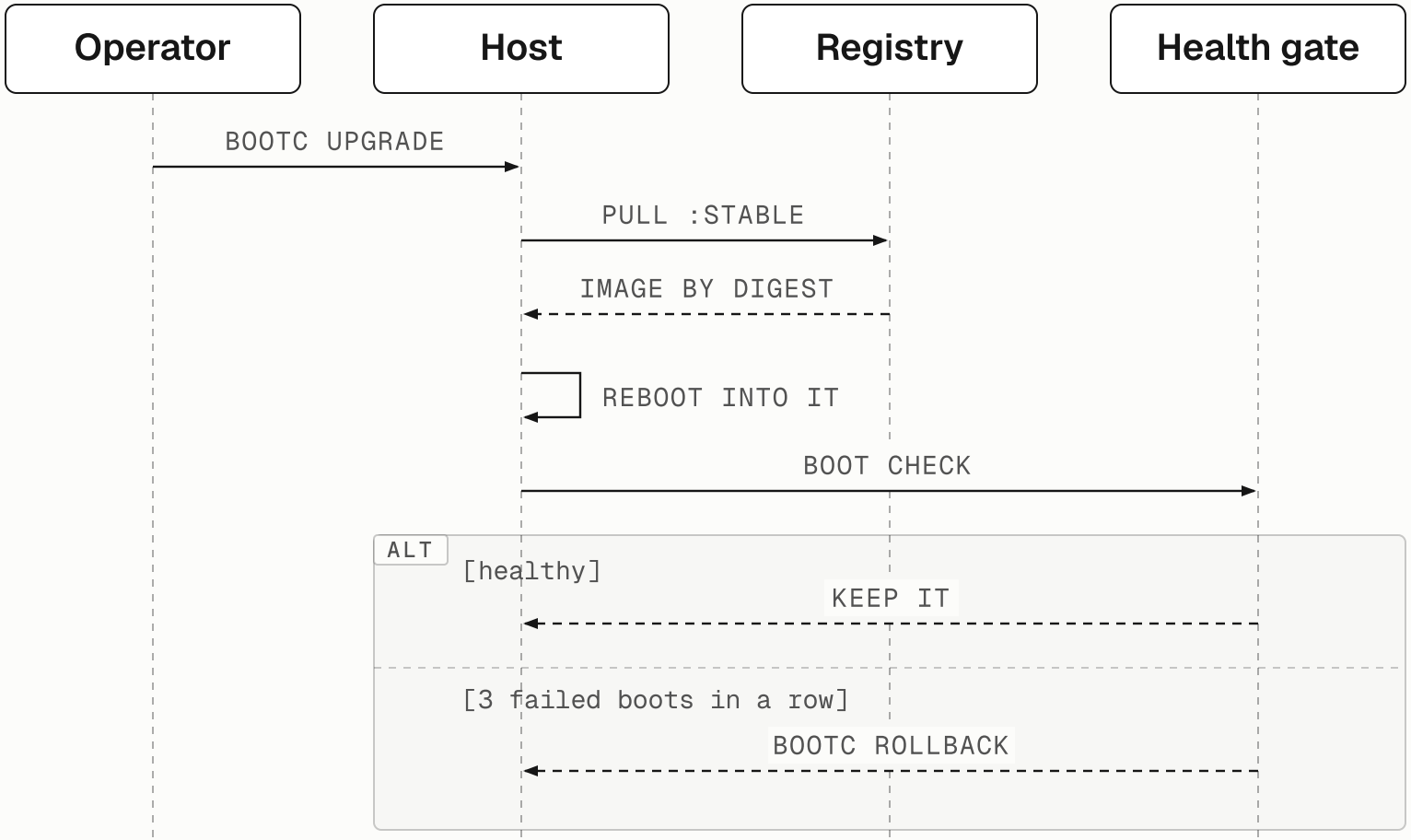


THREE PUBLIC RED RUNS

12 Sep	build	key unset, the guard did not fail closed	pipeline 2843113289
02 Oct	seal	a world-writable file in the sealed tree	pipeline 2905336957
28 Sep	boot	booted, then an assertion did not hold	pipeline 2888347958

Decide what a gate must prove, then break exactly that on purpose.

Updating afterwards



RE-SHIPPED ON A SAME-COMMIT REBUILD

EARLIER RECHUNKER · 418.1 OF 832.4 MIB

50.2%

CHUNKAH 0.7.1 · 43.8 OF 832.5 MIB

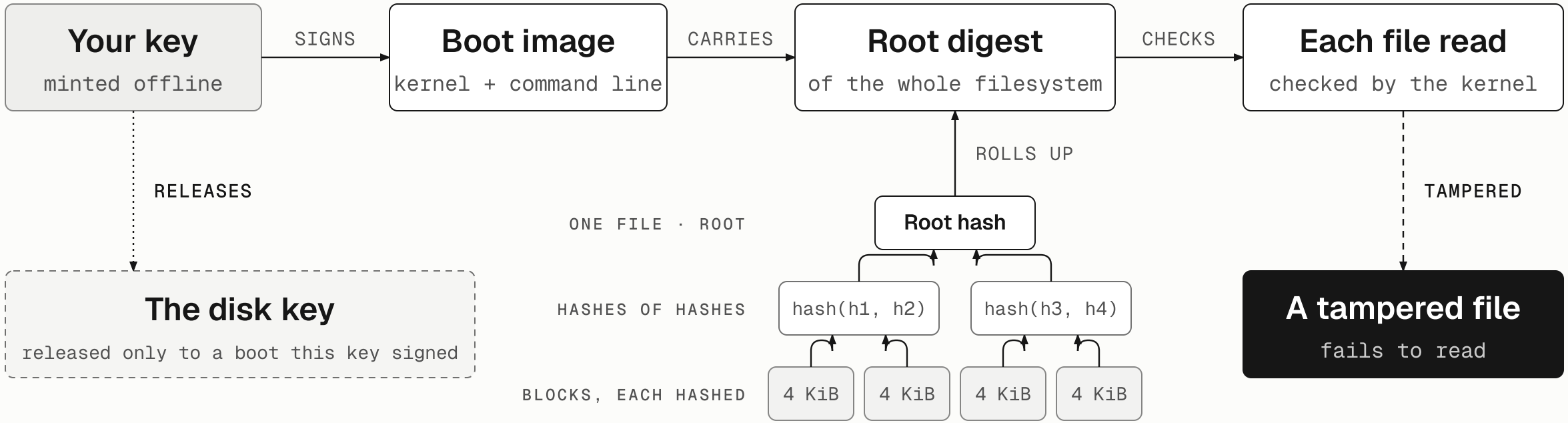
5.3%

About 5% re-shipped on a same-commit rebuild

n = 1 pair of days, 17 to 18 Oct 2026
same commit, flagship base only
host layers on top are not rechunked
a new commit still re-ships the big files

The health gate is a host add-on: it counts failed boots, reboots to retry, and rolls back at the threshold.

What sealed means

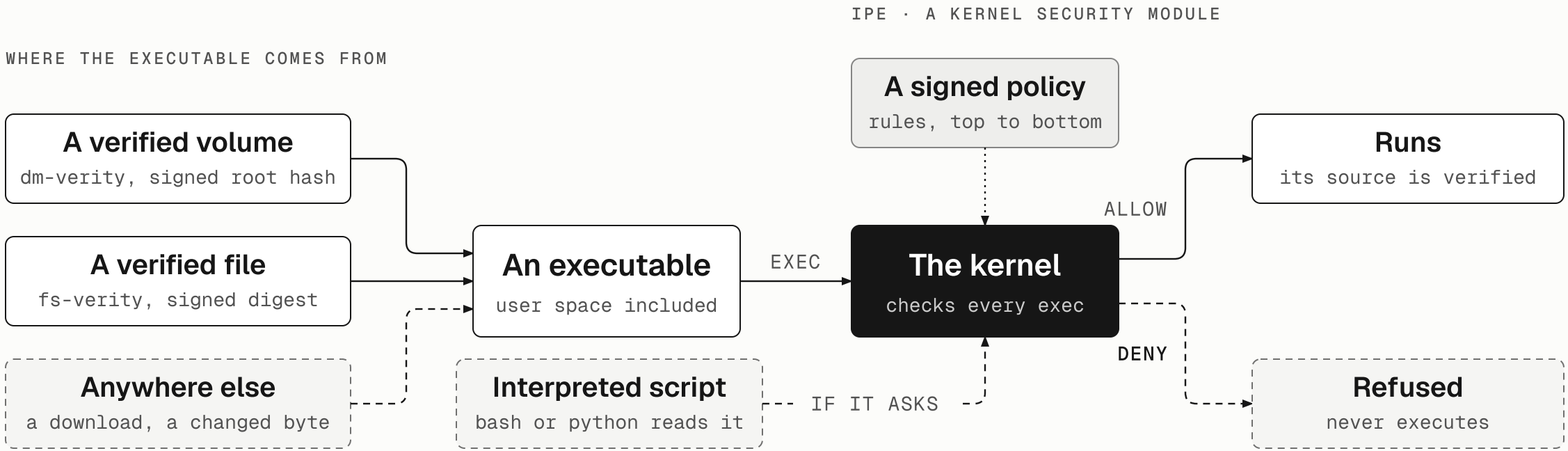


Every file is a tree of hashes. The kernel checks a block against it each time the block is read, not once.

A changed byte fails the read. Wrong key, wrong boot: the host falls back to its recovery passphrase.

fs-verity: docs.kernel.org

The next step: refuse to run what is not verified



UPSTREAM

In the Linux kernel as a security module: it verifies "all executable code before allowing them to run".
docs.kernel.org/admin-guide/LSM/ipe.html

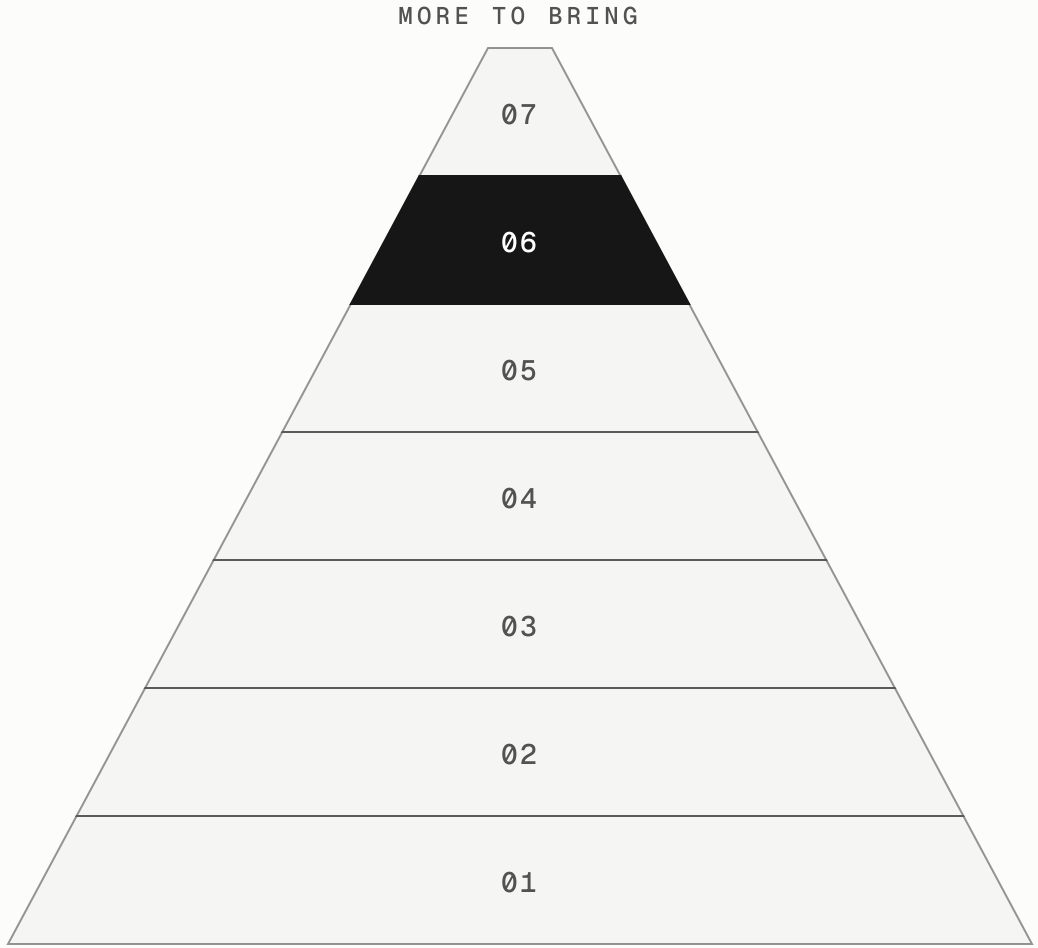
RED HAT

Sealed images verify every OS file on read, a Technology Preview in RHEL 10.2. The composefs project tracks IPE integration: issue 360, opened by a bootc maintainer, open.
redhat.com, sealed images composefs/composefs#360

WHAT I RUN TODAY

The sealed root is verified on read and the kernel modules are signed. IPE is not in what I run.
cairn.dunn.dev/architecture/sealing

How far to go, and what you bring



The example lanes build on the unsealed base.

■ where the house runs today

RUNG · WHAT YOU BRING

Automatic rollback

a health gate that calls bootc rollback

Sealed lane

an offline key hierarchy, a nested-KVM runner, two CI variables

Signature enforcement at pull

a signing keypair

Disk encryption posture

a TPM 2.0 host and an off-device recovery passphrase

A base variant

your hardware facts: filesystem or GPU modules

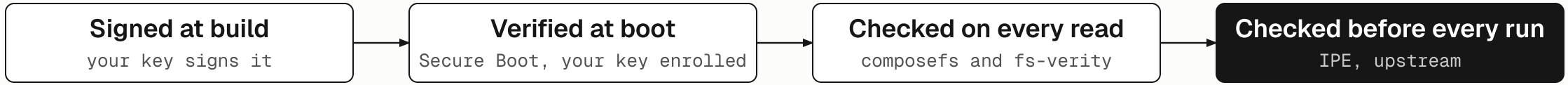
The catalog

one include: build, change-path gate, SBOM, promote

Fork, keep one lane

a registry path and a short Containerfile

Where this is going



RED HAT · LEADING THE WORK

Sealed images

"cryptographically seal your boot container images by using your organization's Secure Boot keys". Technology Preview in RHEL 10.2.

RHEL 10.2 release notes, Technology Preview features

Trust at every layer, from boot to runtime

"The system refuses to load any OS file that does not match."
"This is where OS management is heading."

redhat.com/en/blog, Mark Russell and Colin Walters, 20 May 2026

ALSO WORKING ON VERIFIABLE INTEGRITY

Amutable

LINUX

"Our mission is to deliver verifiable integrity to Linux workloads everywhere": build integrity, boot integrity, runtime integrity.

amutable.com/about

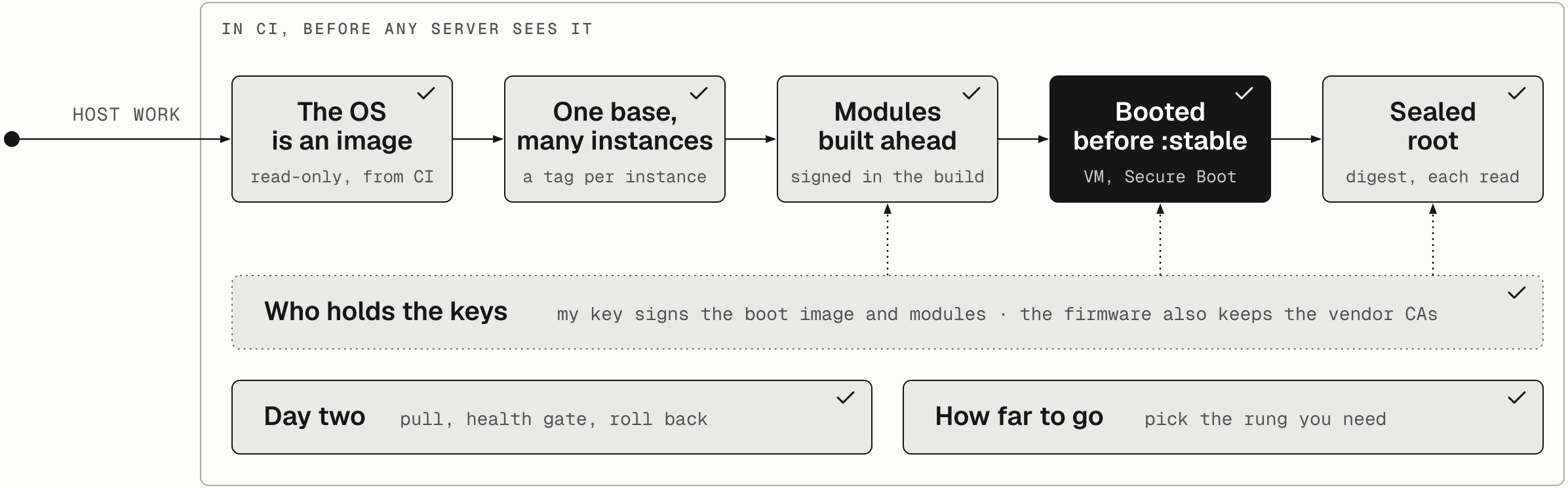
Oxide

THE WHOLE RACK, NOT LINUX

"the switch, the compute sled, and even the power shelf controller have a root of trust", so they "can measure and attest to all of the software".

oxide.computer/faq-friday, root of trust, May 2026

Servers that update like appliances



Your lab: fork the example

	STAGE 1 **Fork**	STAGE 2 **Keep one lane**	STAGE 3 **Your registry**	STAGE 4 **Edit, push**	STAGE 5 **Promoted**
ACTIONS	Fork the example project	Delete the other lanes and their include blocks	Point instance_image at your own registry path	Edit the lane's Containerfile, push to main	Build, change-path gate, SBOM, then :latest and :stable
WHERE	gitlab.com/dunn.dev/cairn/example	lanes/<lane> .gitlab-ci.yml	.gitlab-ci.yml	lanes/<lane> /Containerfile	your registry

NOT A BOOT TEST
needs /dev/kvm on a runner

THREE DOORS



One machine

no CI needed
bootc.dev/bootc



The catalog

gitlab.com/dunn.dev/cairn/pipeline



The example

gitlab.com/dunn.dev/cairn/example

Free to read and use today. GitLab and Red Hat are where you'd put it into practice.

Thanks

WHERE THIS TALK'S IDEA OF SEALING COMES FROM

Mark Russell

Senior Principal Product Manager, Technical, Red Hat

github.com/marrusl

Colin Walters

Engineer at Red Hat, creator of ostree and bootc

github.com/cgwalters

Their post, "Trust at every layer: How sealed images extend OS integrity from boot to runtime", 20 May 2026.

redhat.com/en/blog/how-sealed-images-red-hat-enterprise-linux-extend-os-integrity-boot-runtime

---

Red Hat, for the invitation to give this talk.

Jon Polom, a former DoD colleague I have bounced a lot of these ideas off.

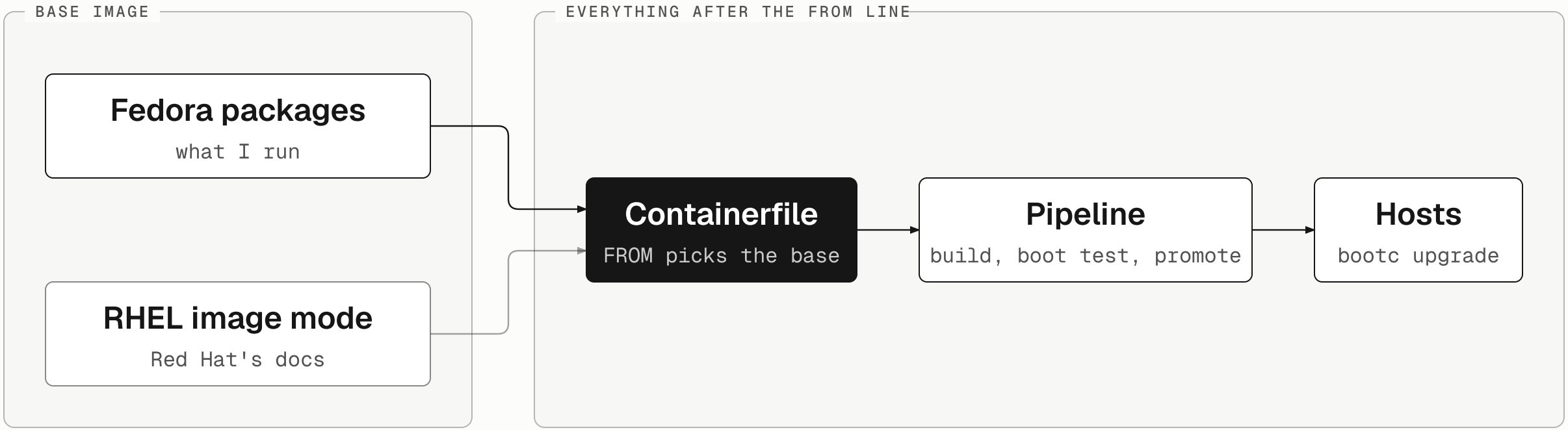
github.com/jmpolom

cairn.dunn.dev andrew.dunn.dev andrew@dunn.dev

Backups

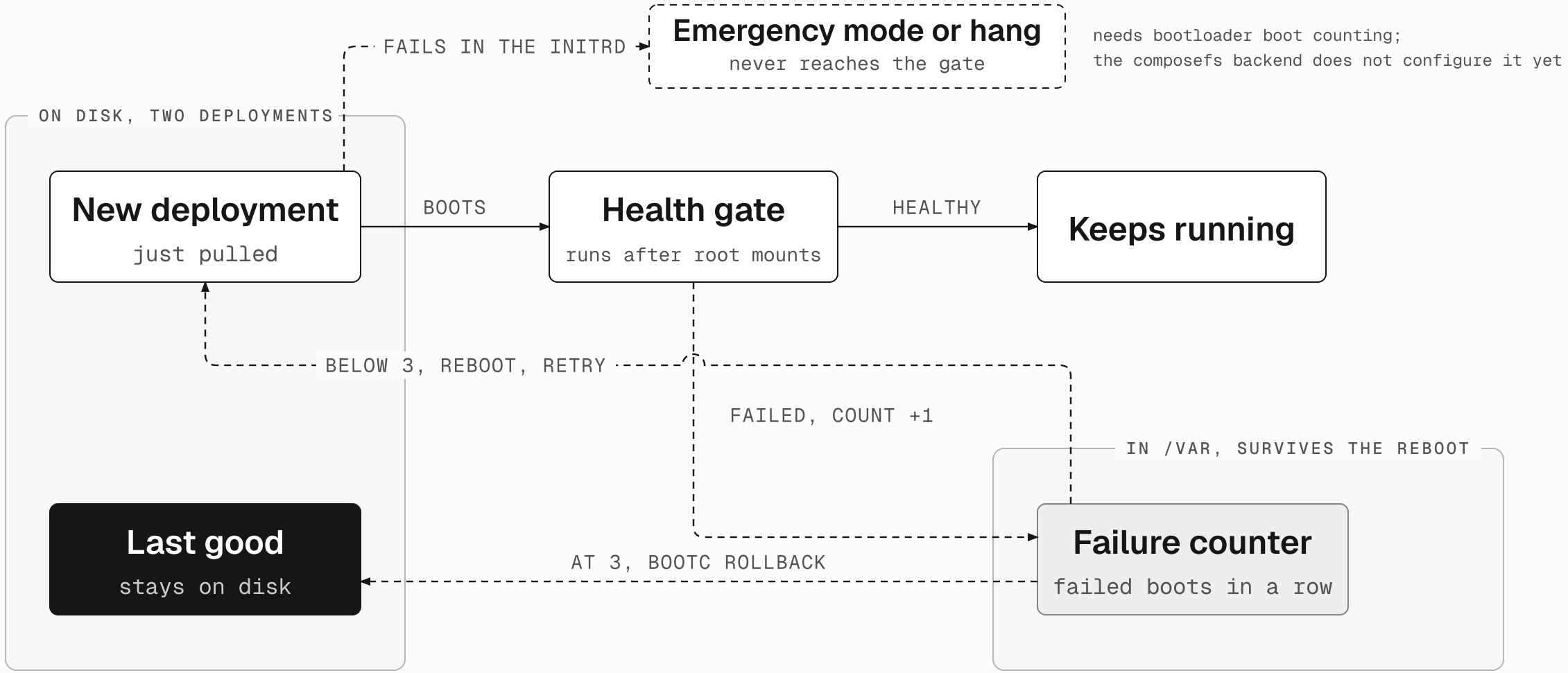
Sixteen faces for the questions I expect. Reached by key or by hash: X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16.

One Containerfile; the FROM line picks the base

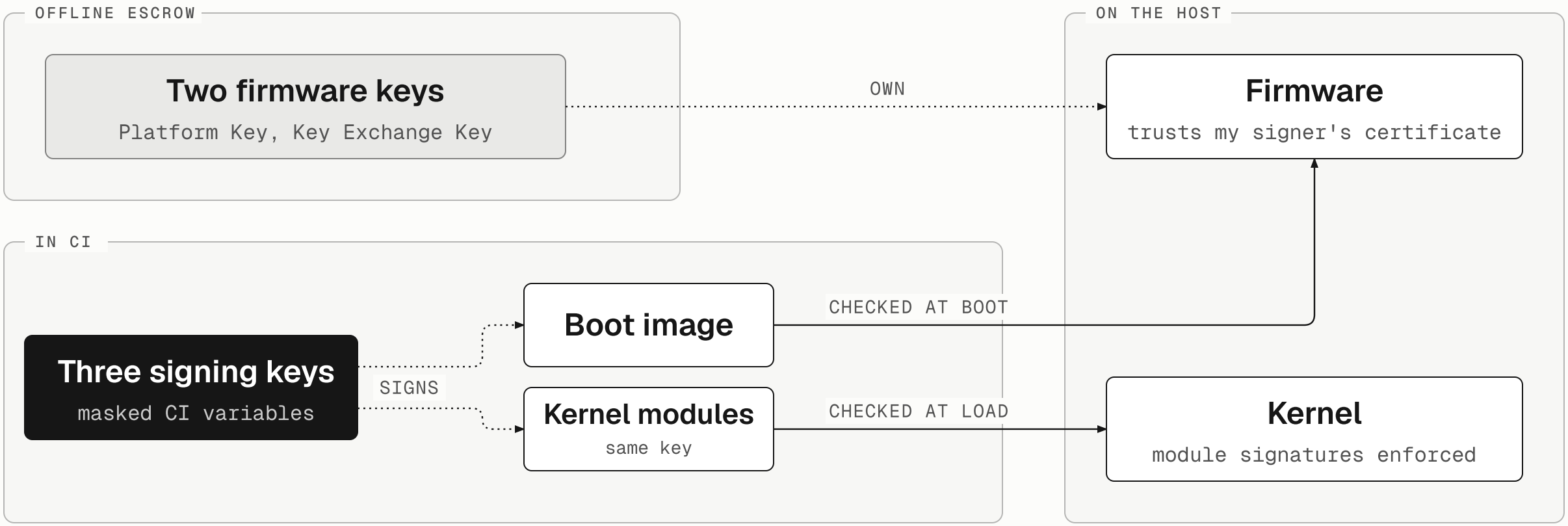


bootc is the same tool on both. Sealing on RHEL is a Technology Preview.

Rollback: the last good deployment stays on disk

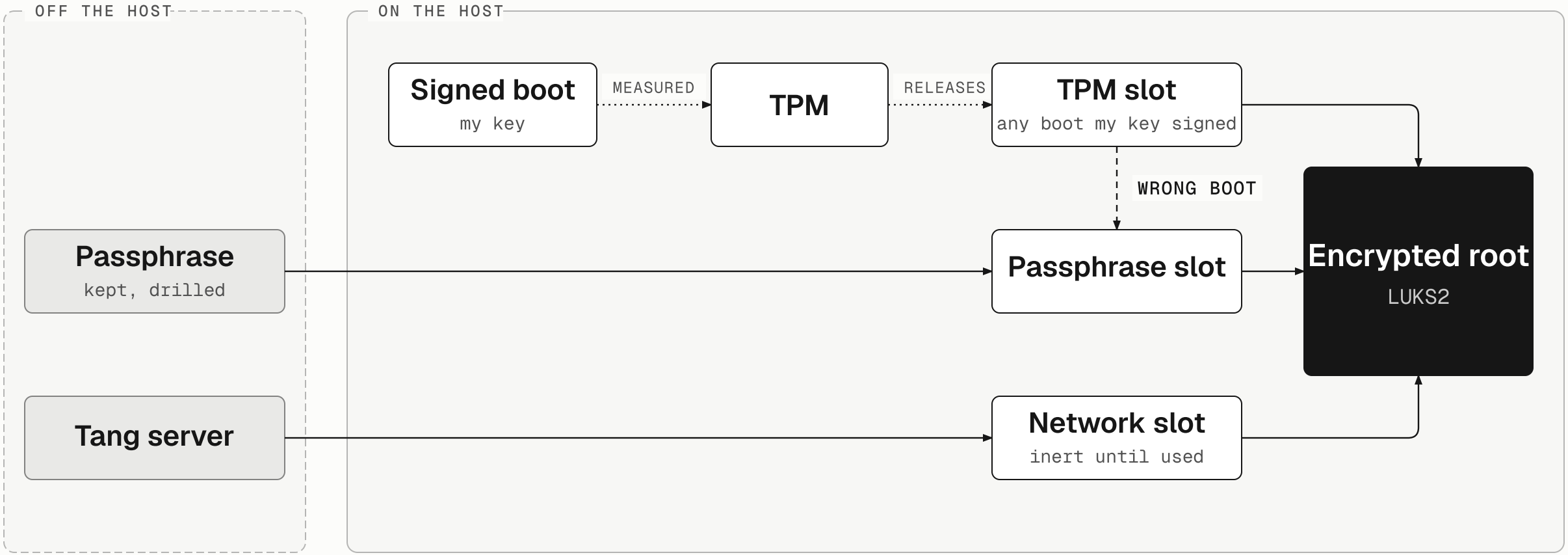


Two keys stay offline; three sign in CI



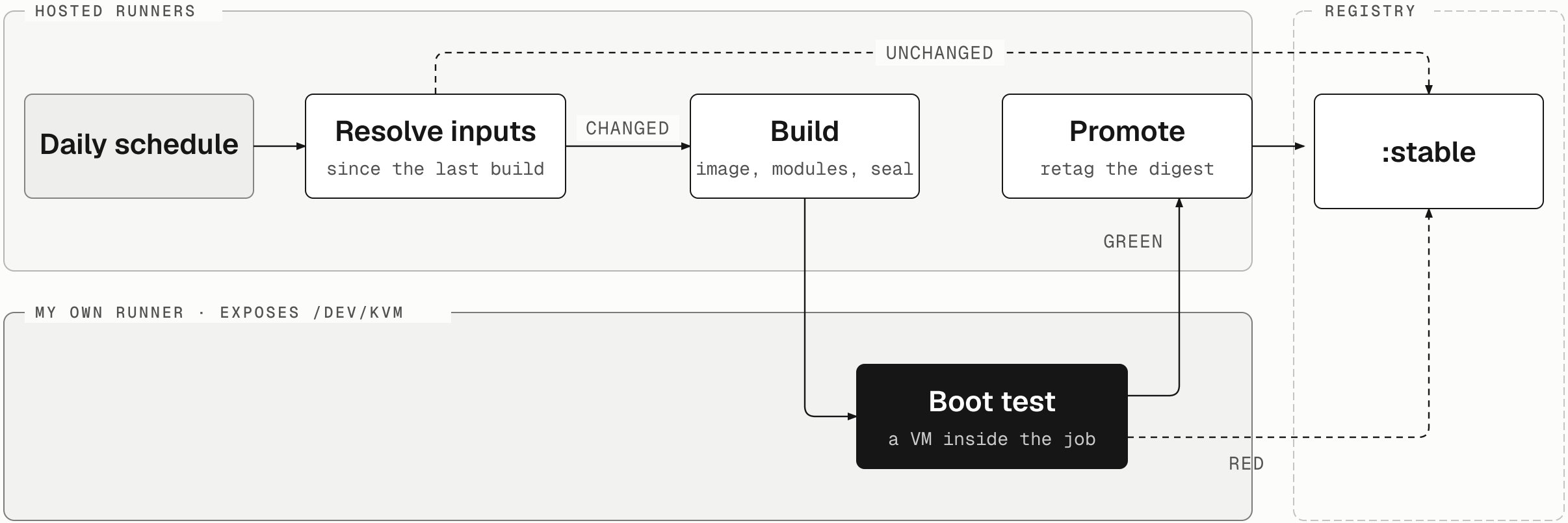
Public halves: cairn.dunn.dev/keys/

Disk unlock: three slots, one bound to the signed boot



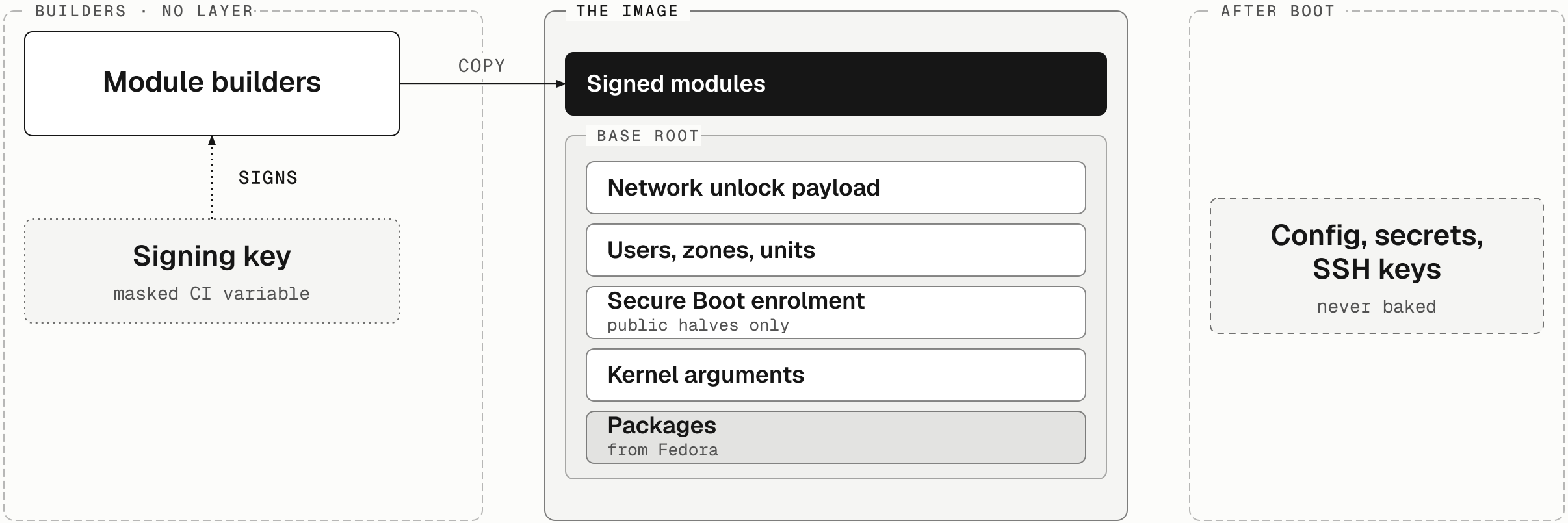
Wrong boot: degraded, not bricked.

Where the work runs, and when none runs



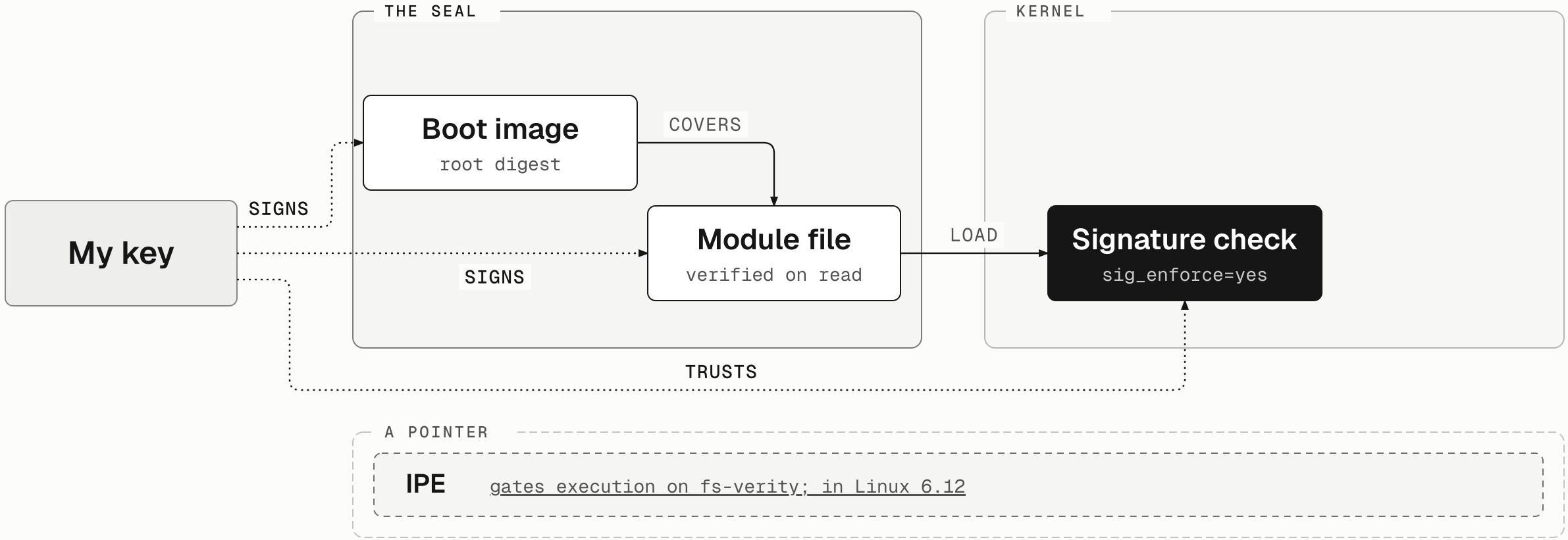
Only the boot test needs a runner of my own.

The Containerfiles, read as layers



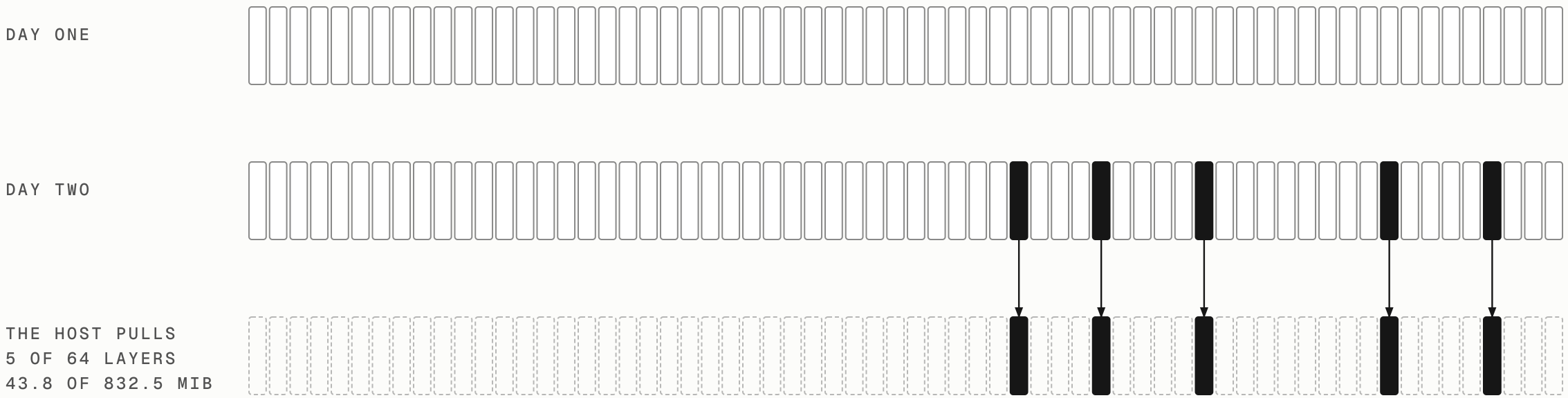
Nothing private lands in a layer.

Where the seal stops and module signing starts



The seal proves the bytes; loading asks who signed them.

Why a same-commit rebuild re-ships about 5%



Identical files make identical layers.

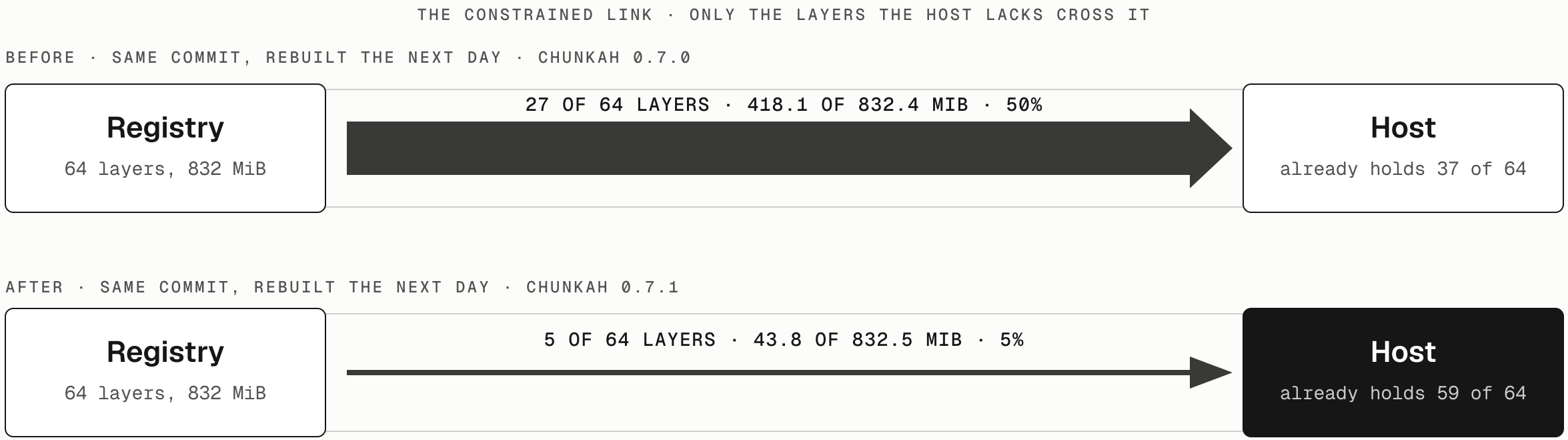
One pair of days, base only. A new commit re-ships the big files.

Re-shipped: small files, driver module, package database, unplaced files, package manager state.

re-shipped already held

X9 · BACKUP

Fewer bytes over a constrained link



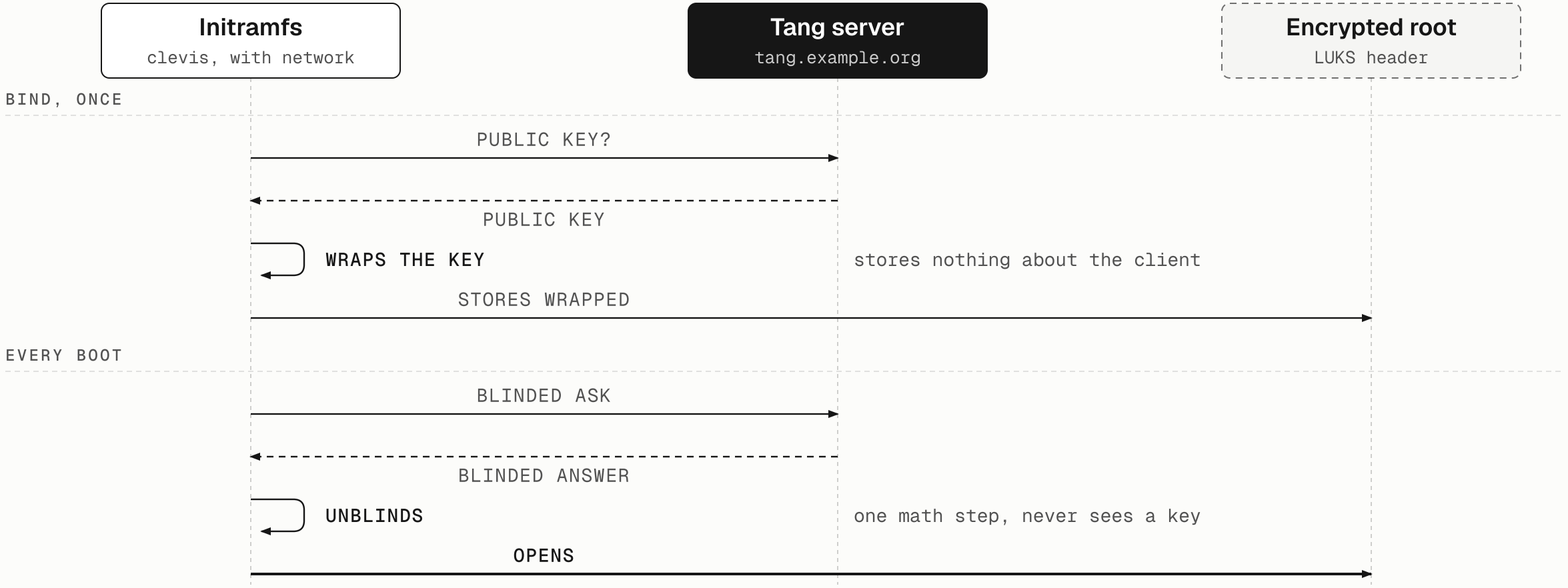
Layers are addressed by digest; a layer the host already holds is not fetched. Rechunking groups files into layers by content, so a layer is byte-identical across days when its files are.

The saving is for rebuilds of unchanged source. A real content change still re-ships the large-file layers.

one image, one pair of days, 17 to 18 Oct 2026 (n = 1) · the host's own layers on top are not rechunked

bytes that cross the link chunkah: github.com/coreos/chunkah bootc: "layers already present; layers needed"

Network-bound disk unlock: the Tang exchange



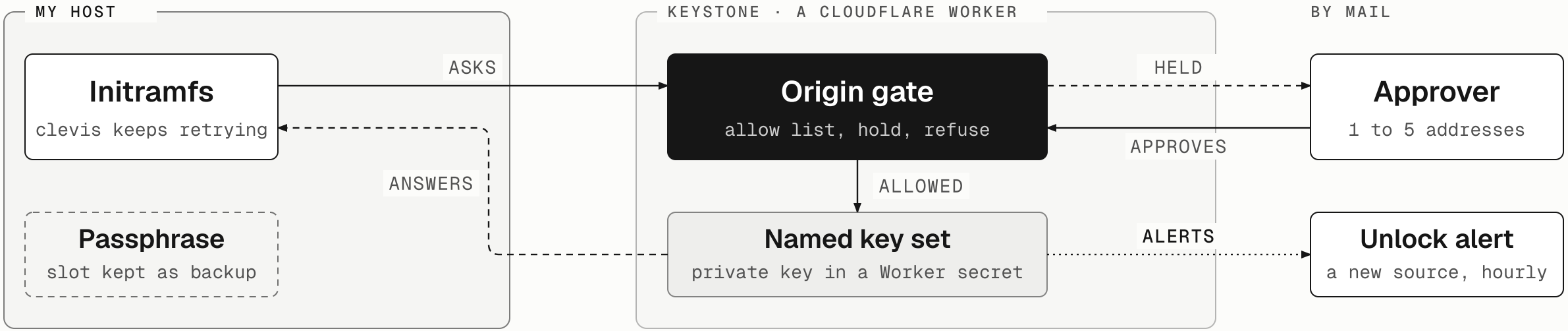
The server holds one private key and stores nothing per client; a stolen disk is useless away from a reachable server.

no TLS, no authentication, blinded both ways · the client's own private half is discarded at bind, only the server can finish the math

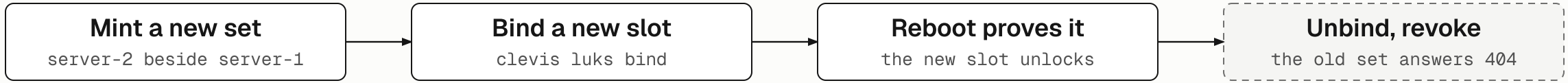
Tang: github.com/latchset/tang

Clevis on RHEL: docs.redhat.com, policy-based decryption

Keystone: a Tang server with an approver in the path



ROTATE BY NAME, THEN REVOKE · KEYS ARE WRITE-ONCE



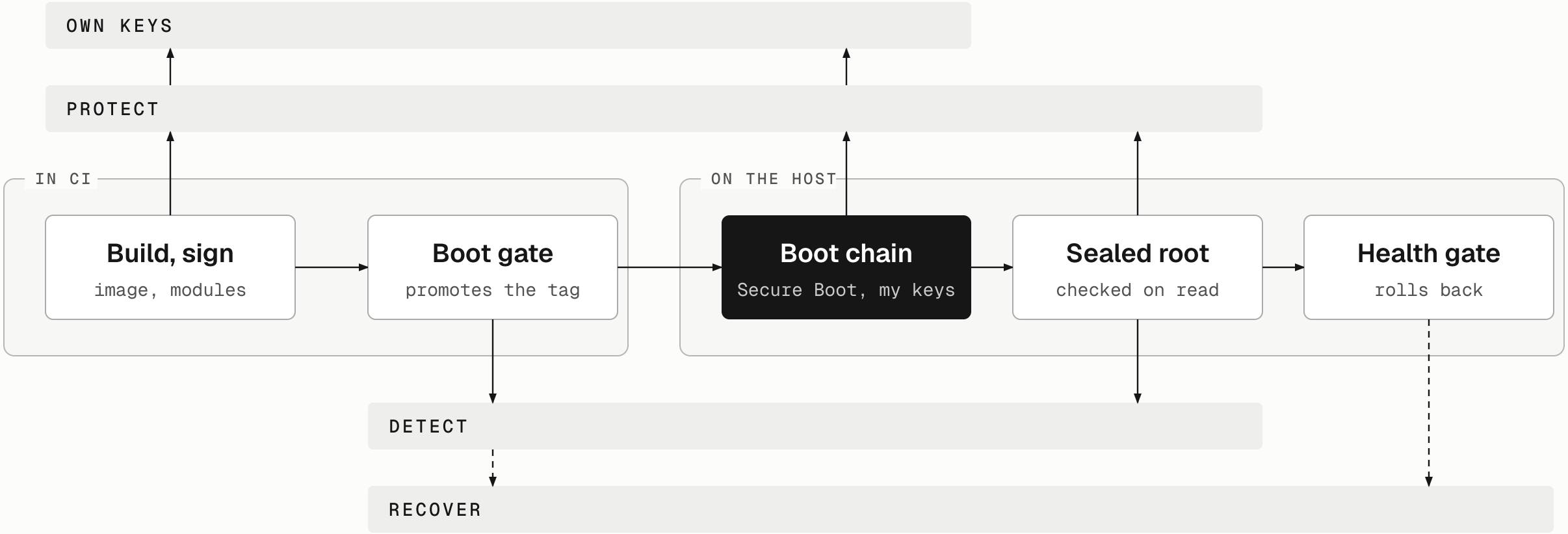
Released, not running on my hosts. A host that already unlocked keeps its disk key; a revoked set stops the next unlock.

whoever controls the account controls the keys; split trust with clevis sss across a second server

an approval serves that prefix for 10 minutes · a one-shot client fails while held; clevis-luks-askpass retries

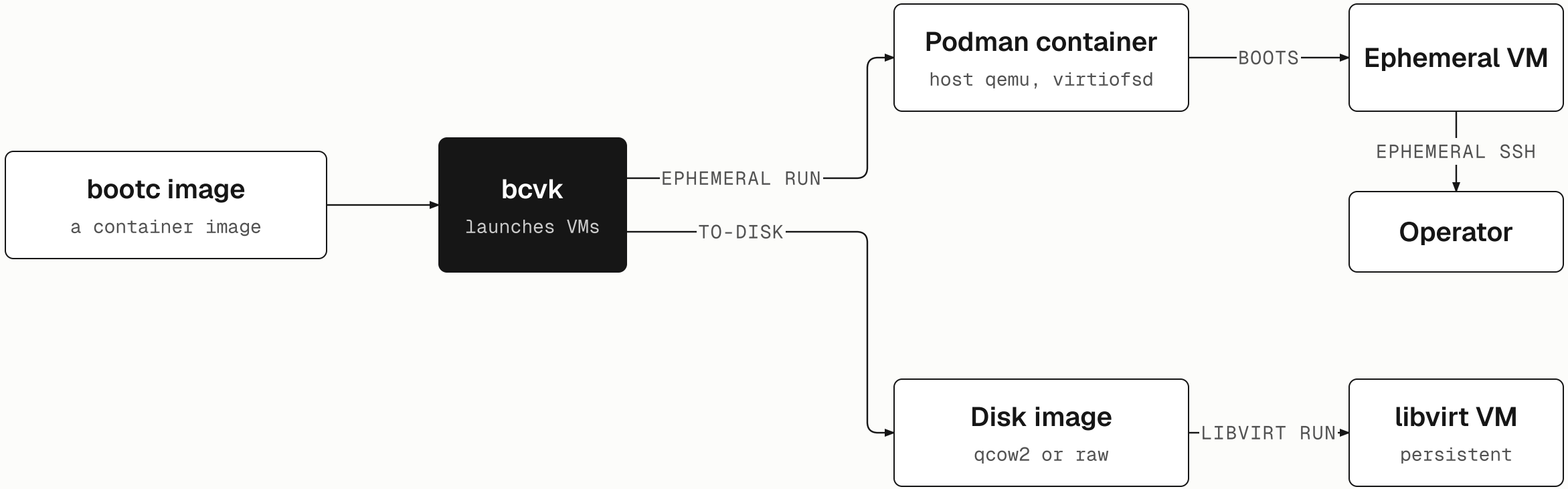
gitlab.com/dunn.dev/keystone keystone.dunn.dev/architecture

Where each function acts



NIST SP 800-193 · NSA, UEFI Secure Boot Customization

bcvk: a bootc image as a virtual machine

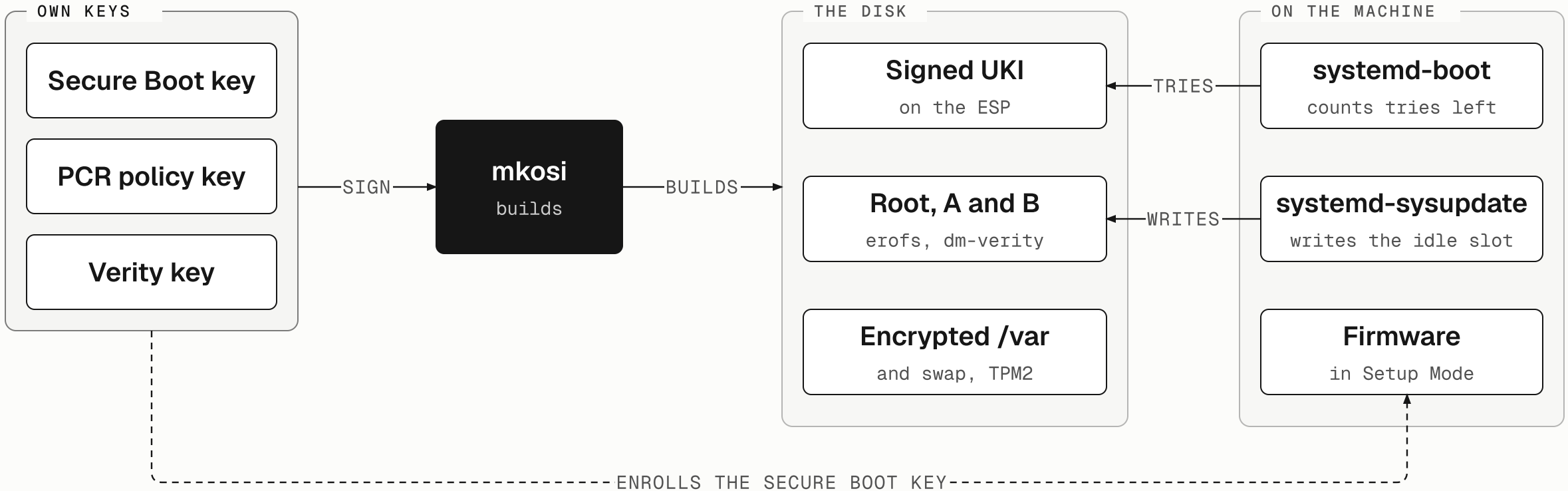


ephemeral needs no root: a wrapper for podman, with qemu and virtiofsd from the host

to-disk runs bootc install to-disk inside an ephemeral VM · ssh keys go in through systemd credentials

github.com/bootc-dev/bcvk RHEL 10 image mode, chapter 10: testing with bcvk

ParticleOS: an image the owner builds and signs

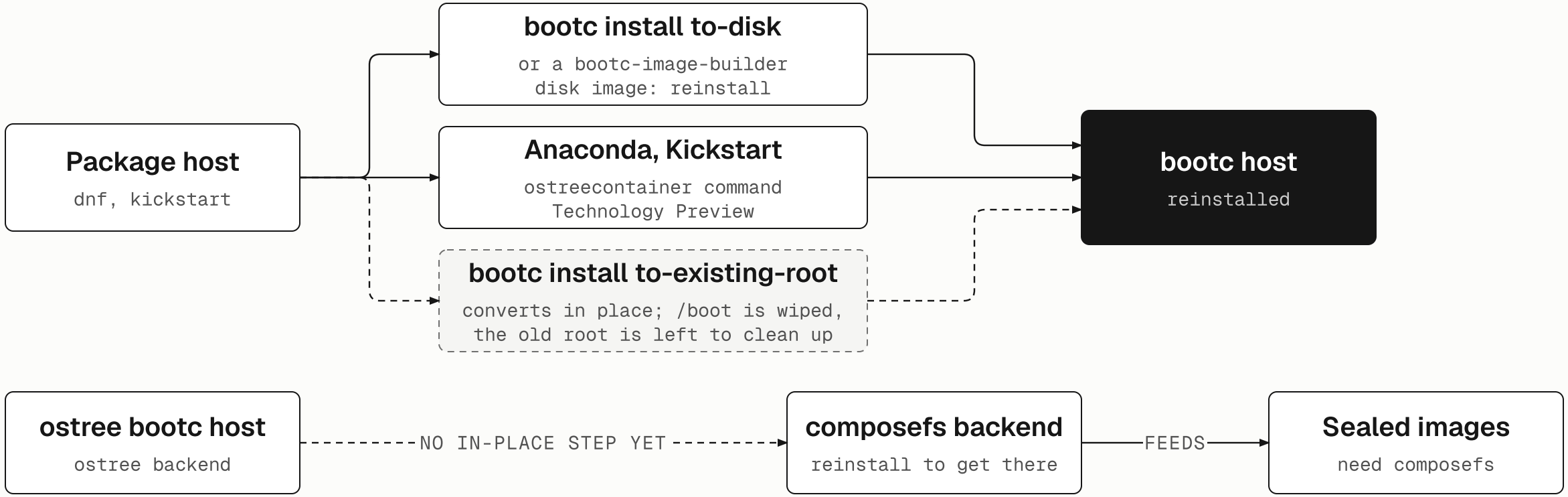


install: the firmware is put in Setup Mode, and a password on its interface keeps Secure Boot on

the FOSDEM 2026 talk lists Secure Boot, dm-verity, the IPE LSM, OBS and systemd-sysupdate among its tooling

github.com/systemd/particleos mkosi root-verity docs FOSDEM 2026 talk

From a package-based fleet to bootc

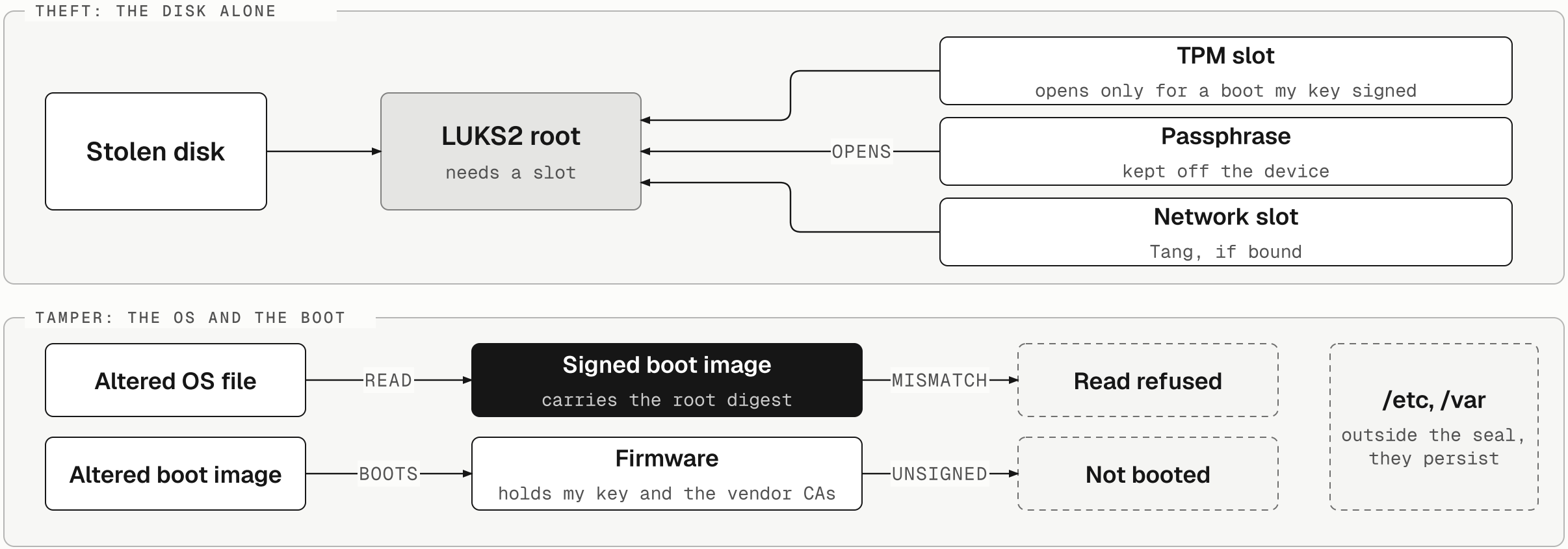


to-existing-root reinitializes /boot, leaves the old root in place · Anaconda, Kickstart, ISO, PXE: Technology Preview

listed as future work: factory reset from ostree to composefs, then copying /etc and /var

bootc composefs bootc install RHEL 10: deploying bootc images

Theft and tamper: where each is stopped



the seal covers the image: one changed byte in the root, and its command-line digest no longer matches

ParticleOS's install note and the NSA customization guide both advise a password on the firmware interface

cairn.dunn.dev/architecture/sealing Red Hat, sealed images NSA, UEFI Secure Boot customization

Credits

Diagram grammar: diagram-design, MIT, Copyright (c) Cathryn Lavery.

Type: Geist and Geist Mono, Vercel, SIL Open Font License.

The GitLab logo is a trademark of GitLab Inc.; the Red Hat logo is a trademark of Red Hat, Inc. Used to name the user group's hosts and the tools discussed.

Sources: source.android.com, support.apple.com, bootc.dev, docs.redhat.com, csrc.nist.gov, NSA UEFI Secure Boot Customization, cairn.dunn.dev and the public cairn repos at gitlab.com/dunn.dev/cairn.

PDF of this deck