"Signed, Sealed, and Delivered", with UKIs and composefs

Allison Karlitskaya
Engineer, Red Hat

Timothée Ravier
CoreOS engineer

Signed, Sealed, and Delivered

Etymology (English)

From an old English common law rule that property was not conveyed from one party to another until the document memorializing the conveyance had been signed by the conveyor, affixed with his seal, and delivered to the recipient of the property.

https://en.wiktionary.org/wiki/signed,_sealed,_and_delivered

Goal?

Create a full chain of trust from the firmware to the root filesystem.

What is composefs?

What is composefs?

It doesn't exist.

A (very) short history

*** [PATCH v3 0/6] Composefs: an opportunistically sharing verified image filesystem**

@ 2023-01-20 15:23 Alexander Larsson

0 siblings, 7 replies; 80+ messages in thread

TL;DR: “No.”

(or)

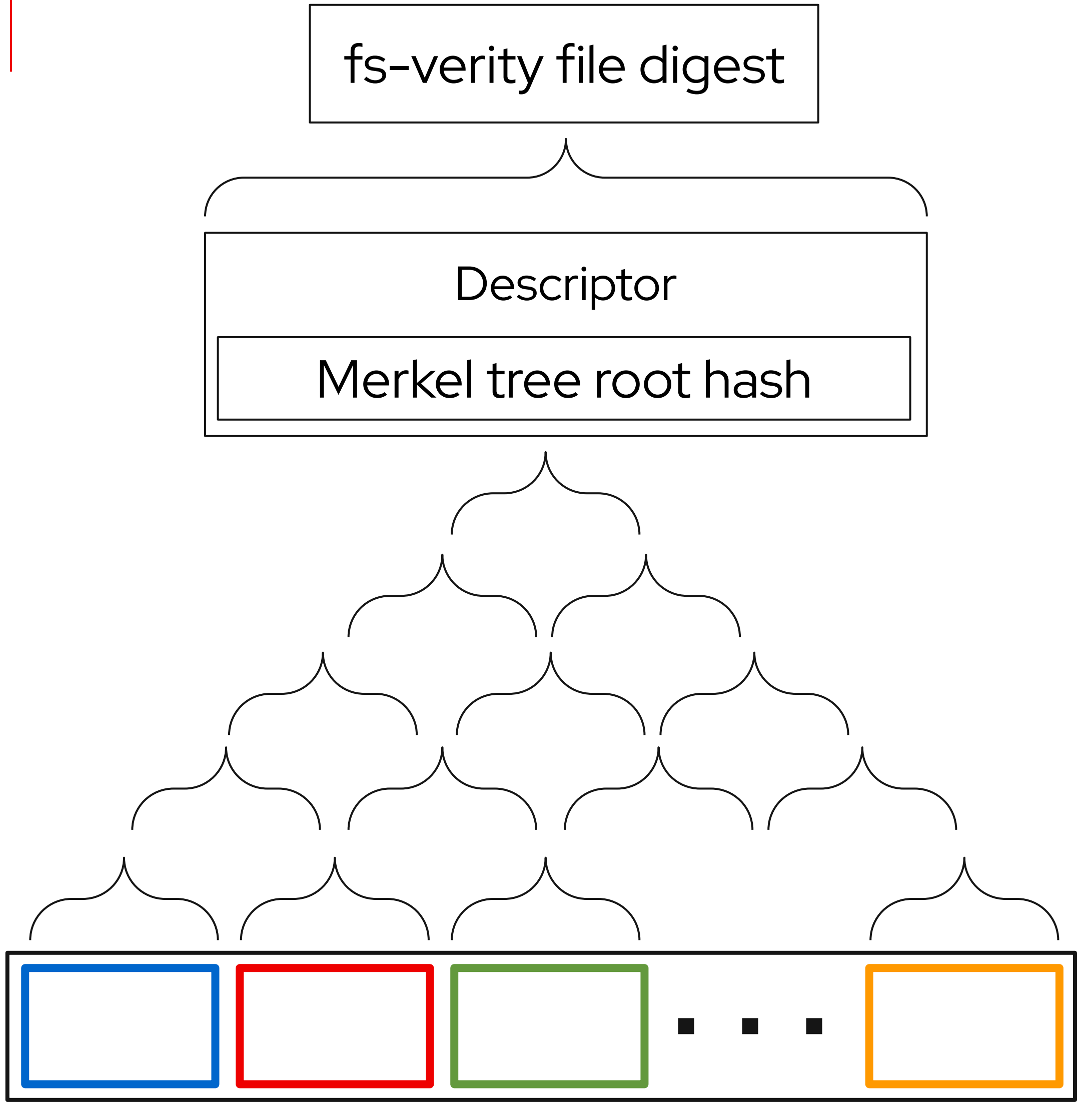
“You can (almost) do this with fs-verity, erofs, and overlayfs.”

What is fs-verity?

A way to assert integrity of file contents.

Doesn’t do directories, doesn’t do file metadata: only file content of regular files.

Supported by Btrfs, ext4, F2FS, and more filesystems are coming.



What is fs-verity?

Merkel tree with a descriptor at the root.

fs-verity is enabled per-file, creating the Merkel tree and storing it in the filesystem.

Files are immutable and can be “measured” to a digest.

File content is verified as it’s read into the page cache.

What is EROFS?

“Yet another read-only filesystem”

Supports all POSIX features.

Actively developed, good communication with upstream.

Performant (including several performance tweaks added for the benefit of composefs, such as bloom filter on xattr lookup).

What is overlayfs?

OverlayFS is a union mount filesystem implementation for Linux. It combines multiple different underlying mount points into one, resulting in single directory structure that contains underlying files and sub-directories from all sources.

https://en.wikipedia.org/wiki/OverlayFS

Important features of overlayfs for large files

How can you erase a file?

- whiteout (character device 0, 0)

How can you rename a large file efficiently?

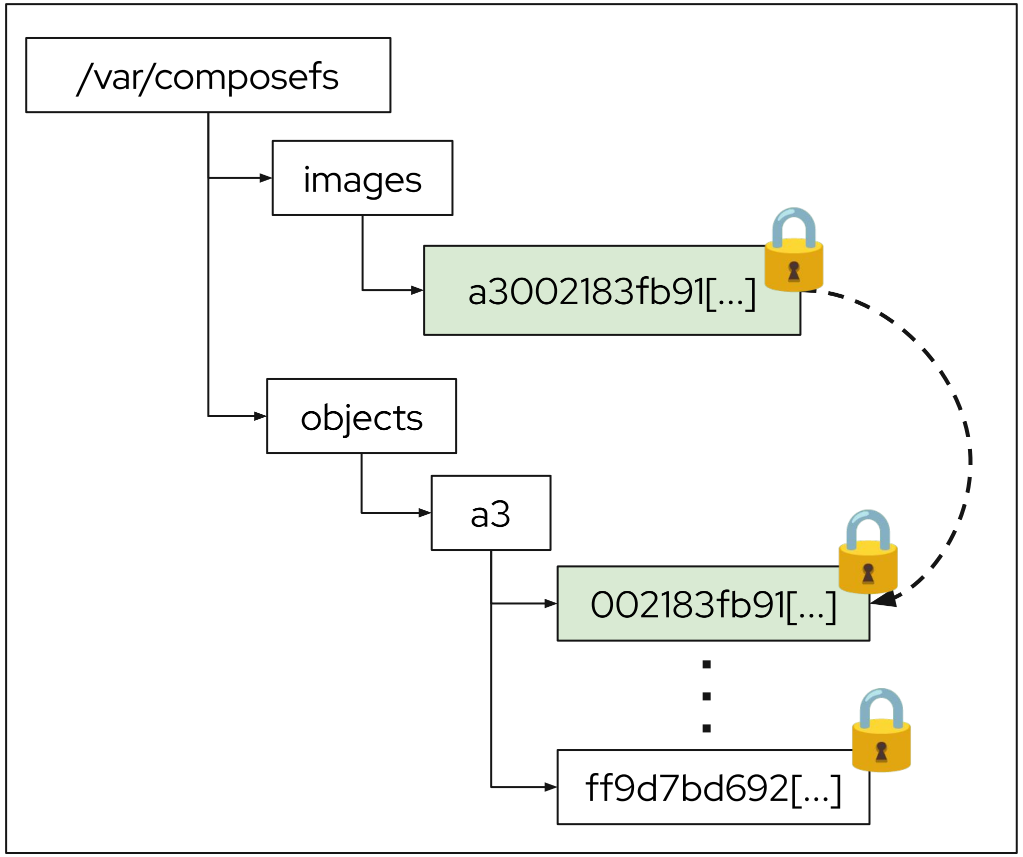
- trusted.overlay.redirect xattr

How can you chmod a large file efficiently?

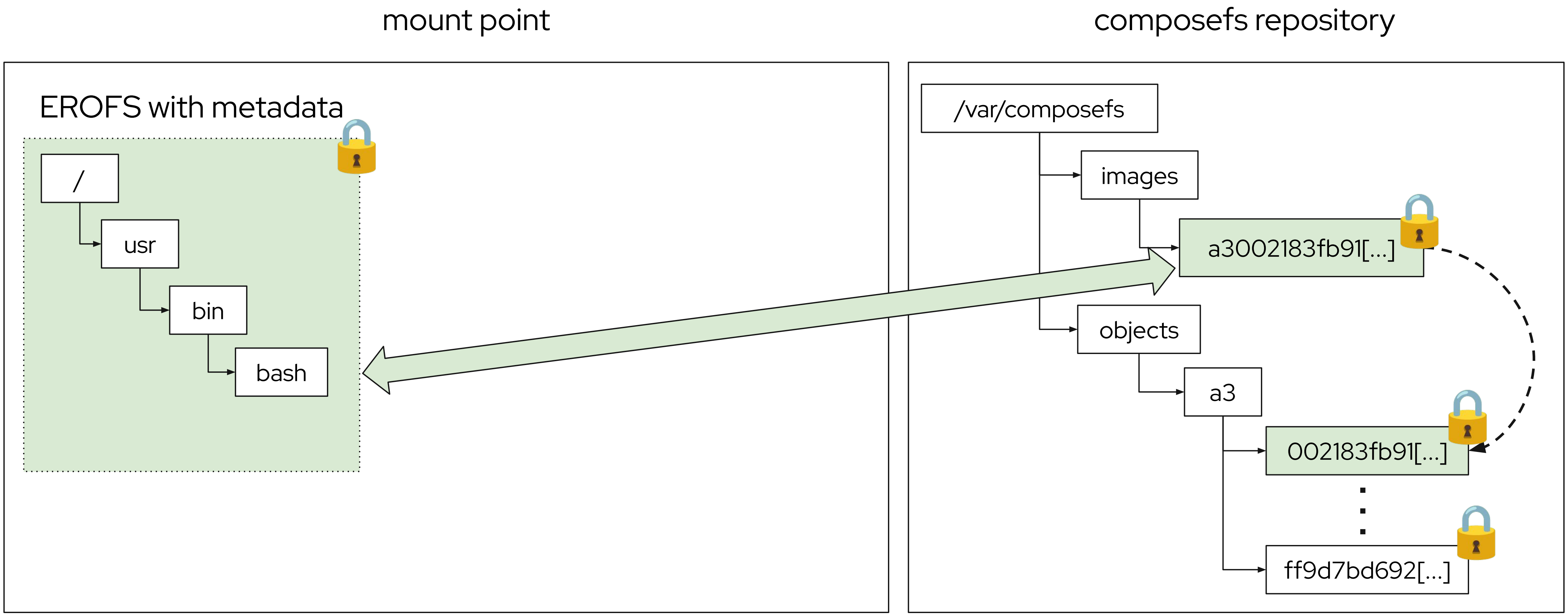
- trusted.overlay.metacopy xattr
- fs-verity verification (only used by composefs)

So then, what is composefs?

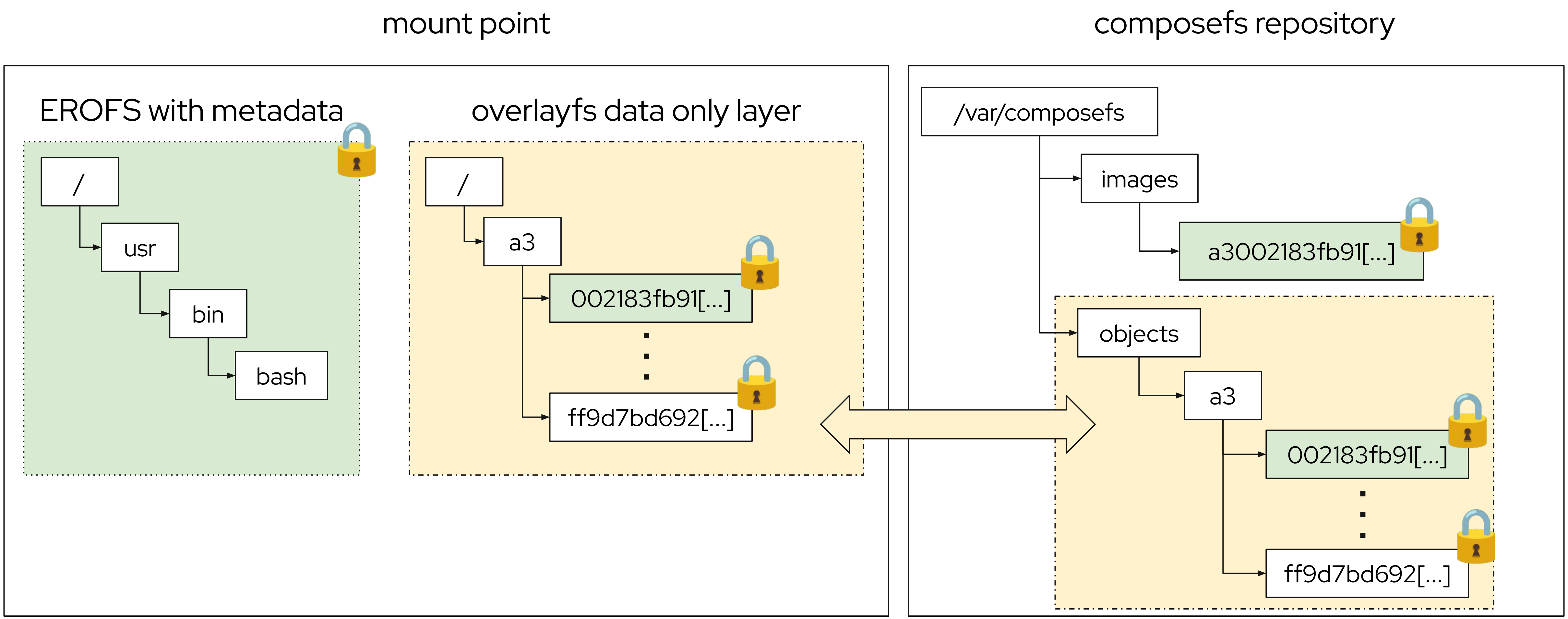
composefs repository



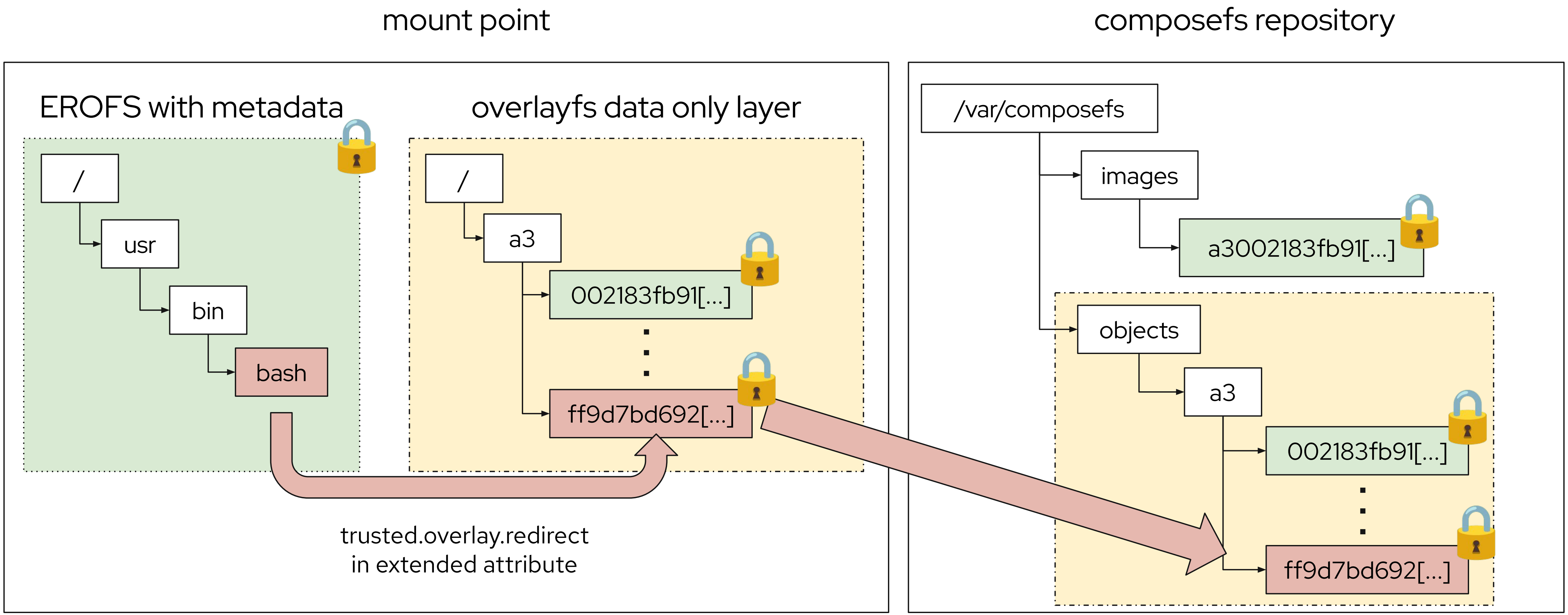
So then, what is composefs?



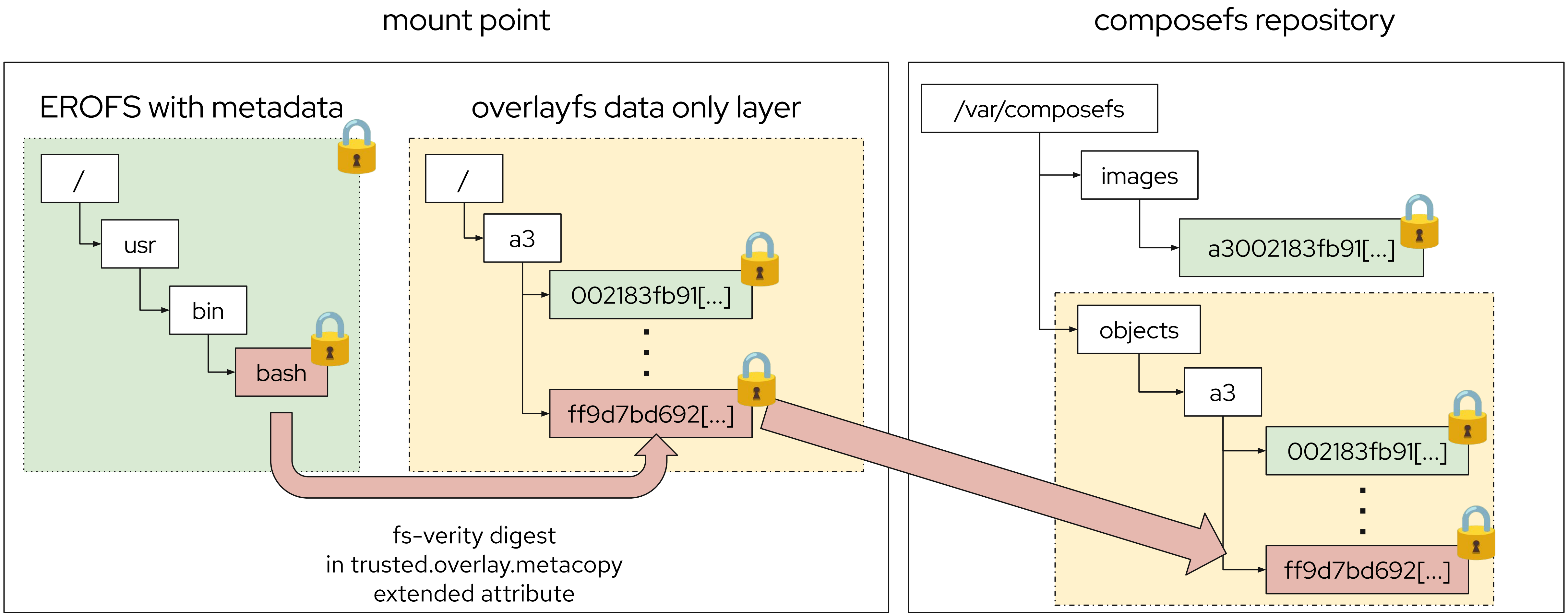
So then, what is composefs?



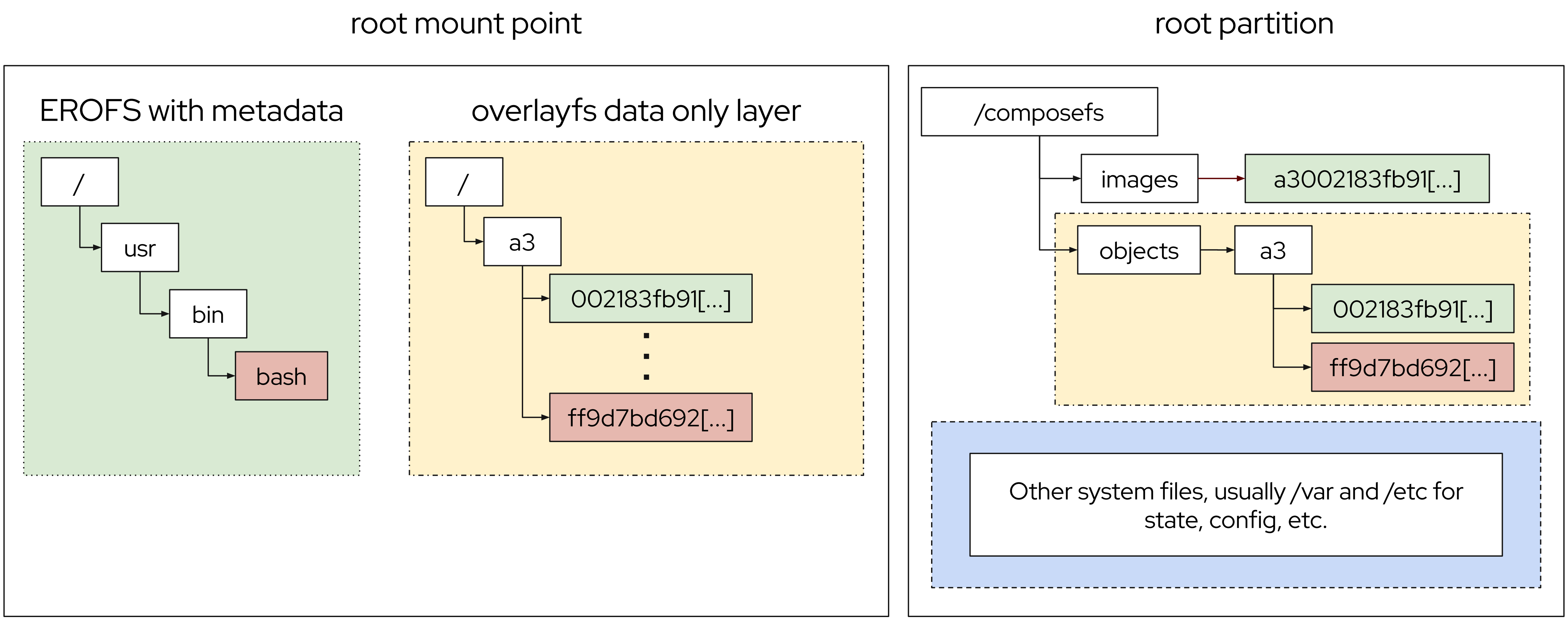
So then, what is composefs?



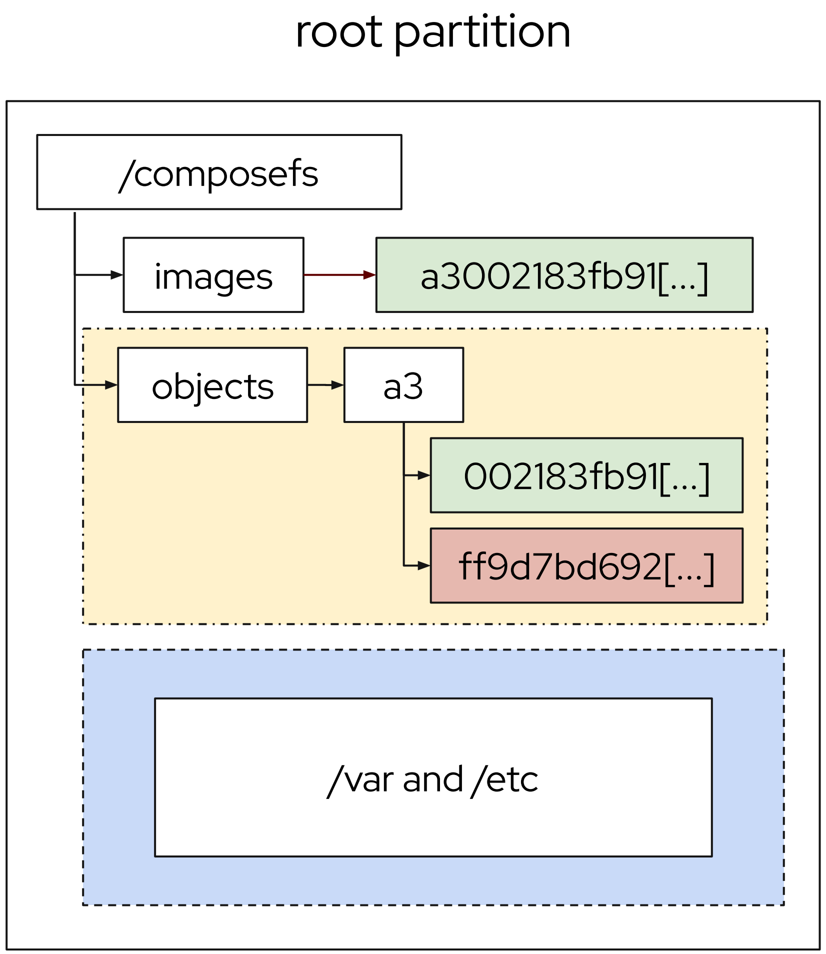
So then, what is composefs?



So then, what is composefs?

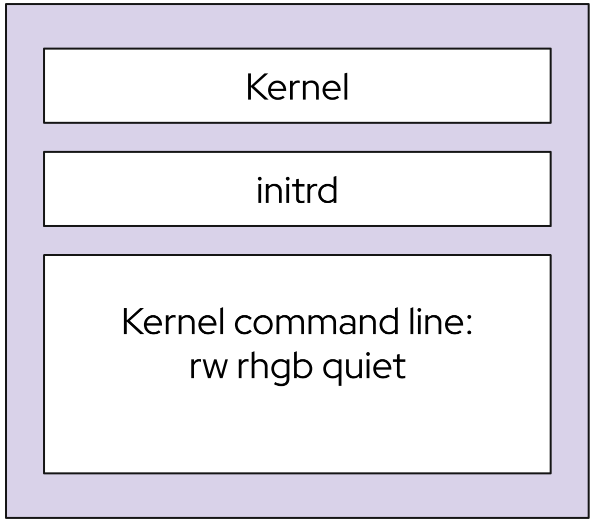


Building a fully verified boot chain



Building a fully verified boot chain

UKI (Unified Kernel Image)

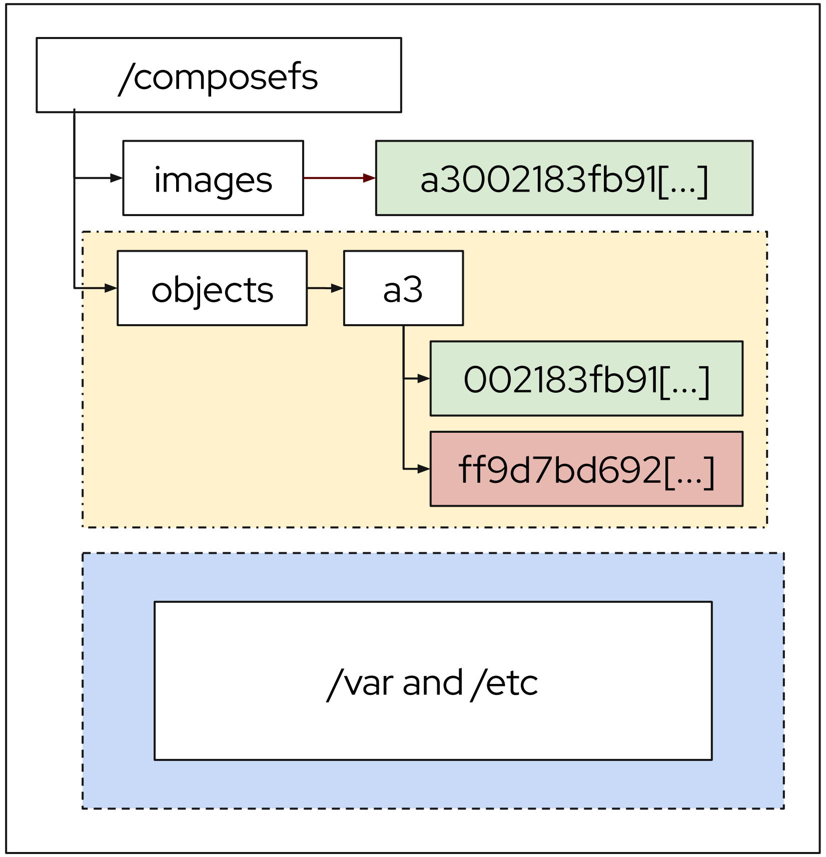


Kernel

initrd

Kernel command line:
rw rhgb quiet

root partition



/composefs

images

a3002183fb91[...]

objects

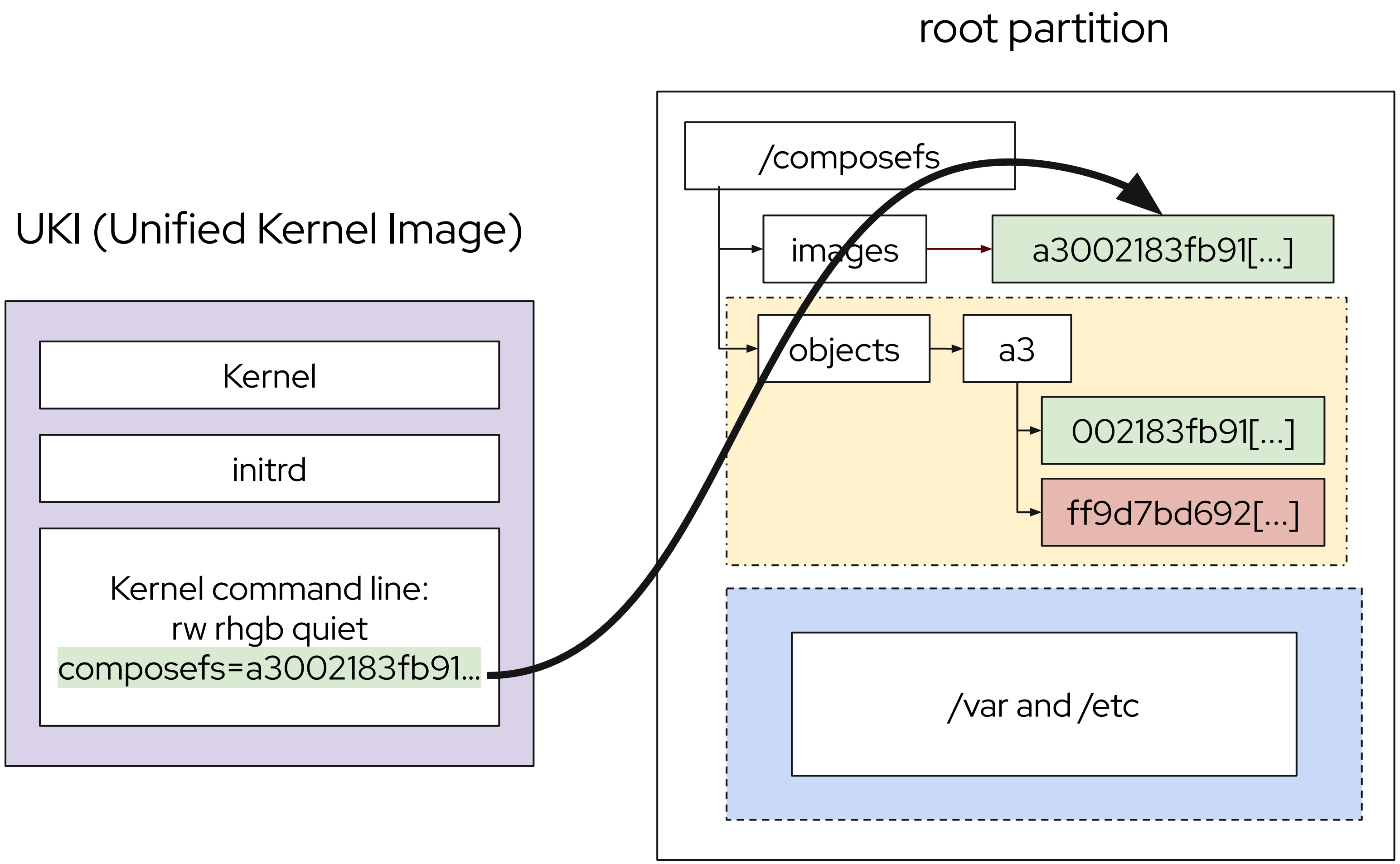
a3

002183fb91[...]

ff9d7bd692[...]

/var and /etc

Building a fully verified boot chain



UKI (Unified Kernel Image)

Kernel

initrd

Kernel command line:
rw rhgb quiet
composefs=a3002183fb91…

root partition

/composefs

images

a3002183fb91[...]

objects

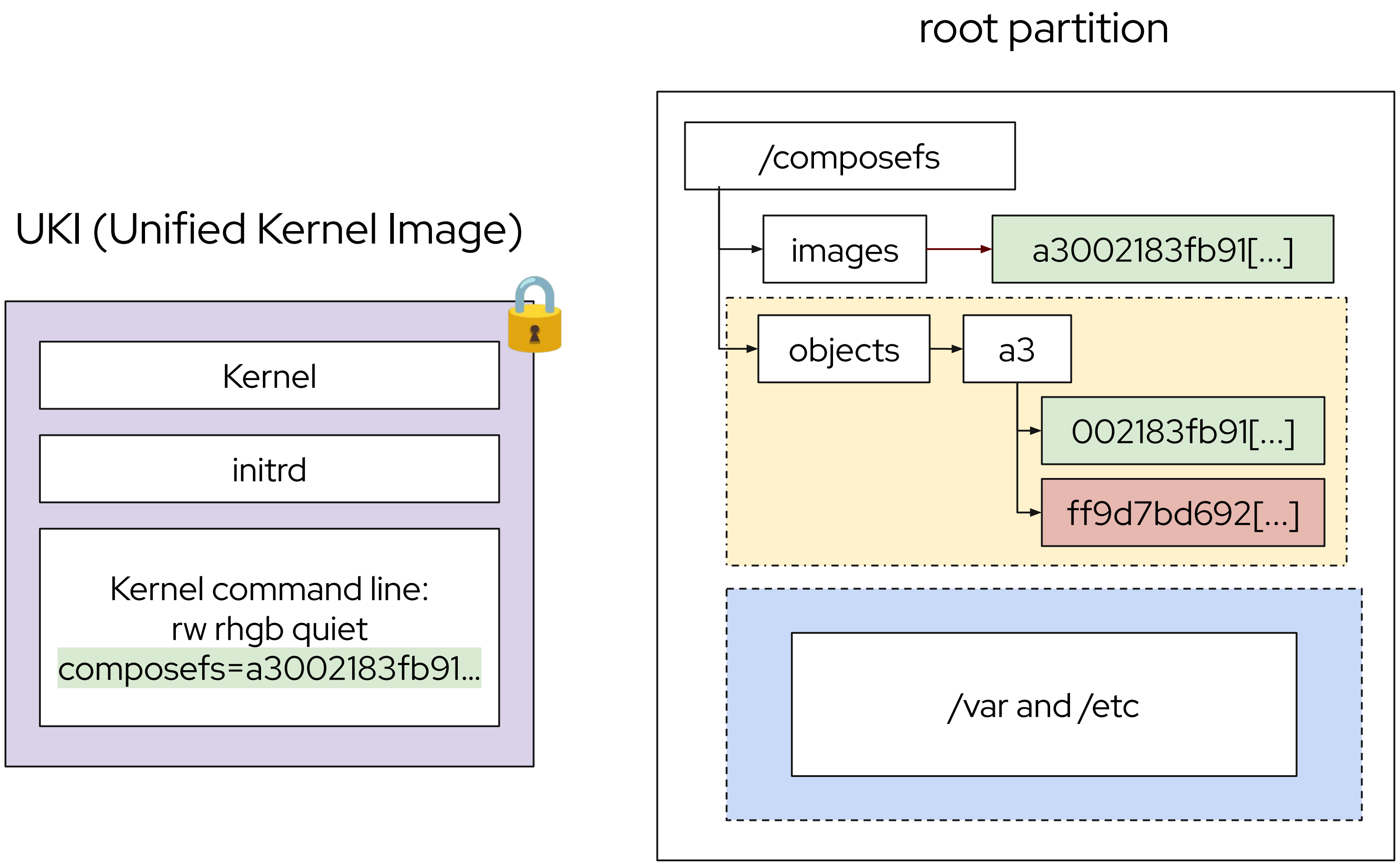
a3

002183fb91[...]

ff9d7bd692[...]

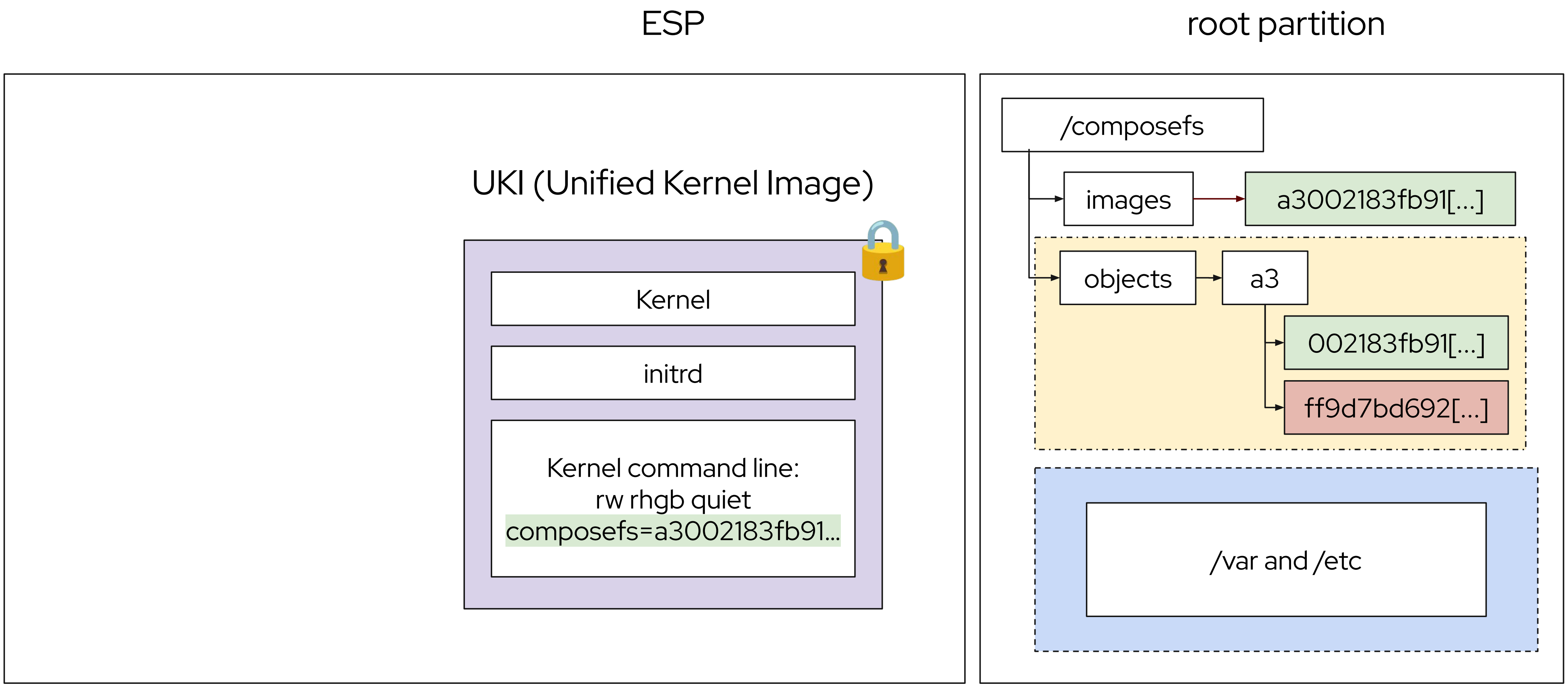
/var and /etc

Building a fully verified boot chain



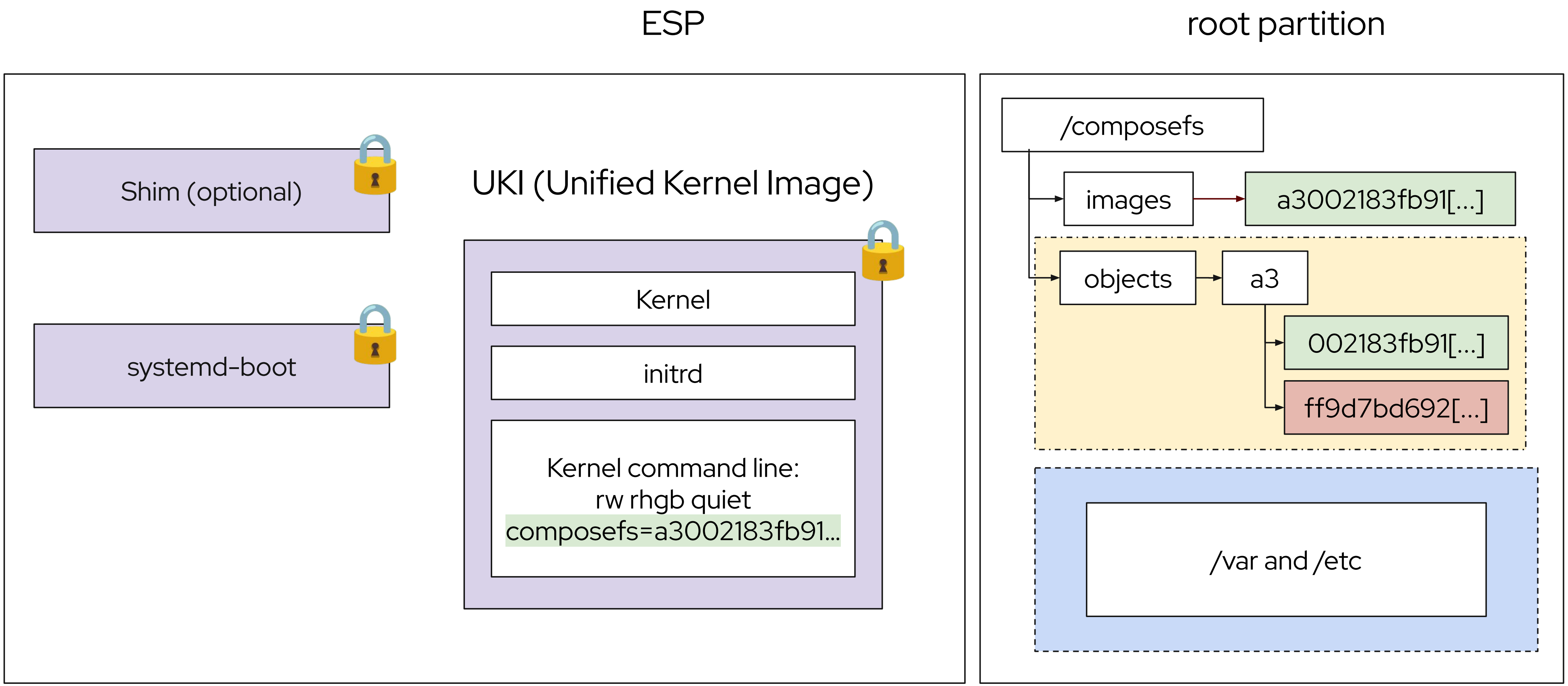
🔒 Signed for Secure Boot

Building a fully verified boot chain



🔒 Signed for Secure Boot

Building a fully verified boot chain



ESP

Shim (optional) 🔒

systemd-boot 🔒

UKI (Unified Kernel Image) 🔒

Kernel

initrd

Kernel command line:
rw rhgb quiet
composefs=a3002183fb91...

root partition

/composefs

images

a3002183fb91[...]

objects

a3

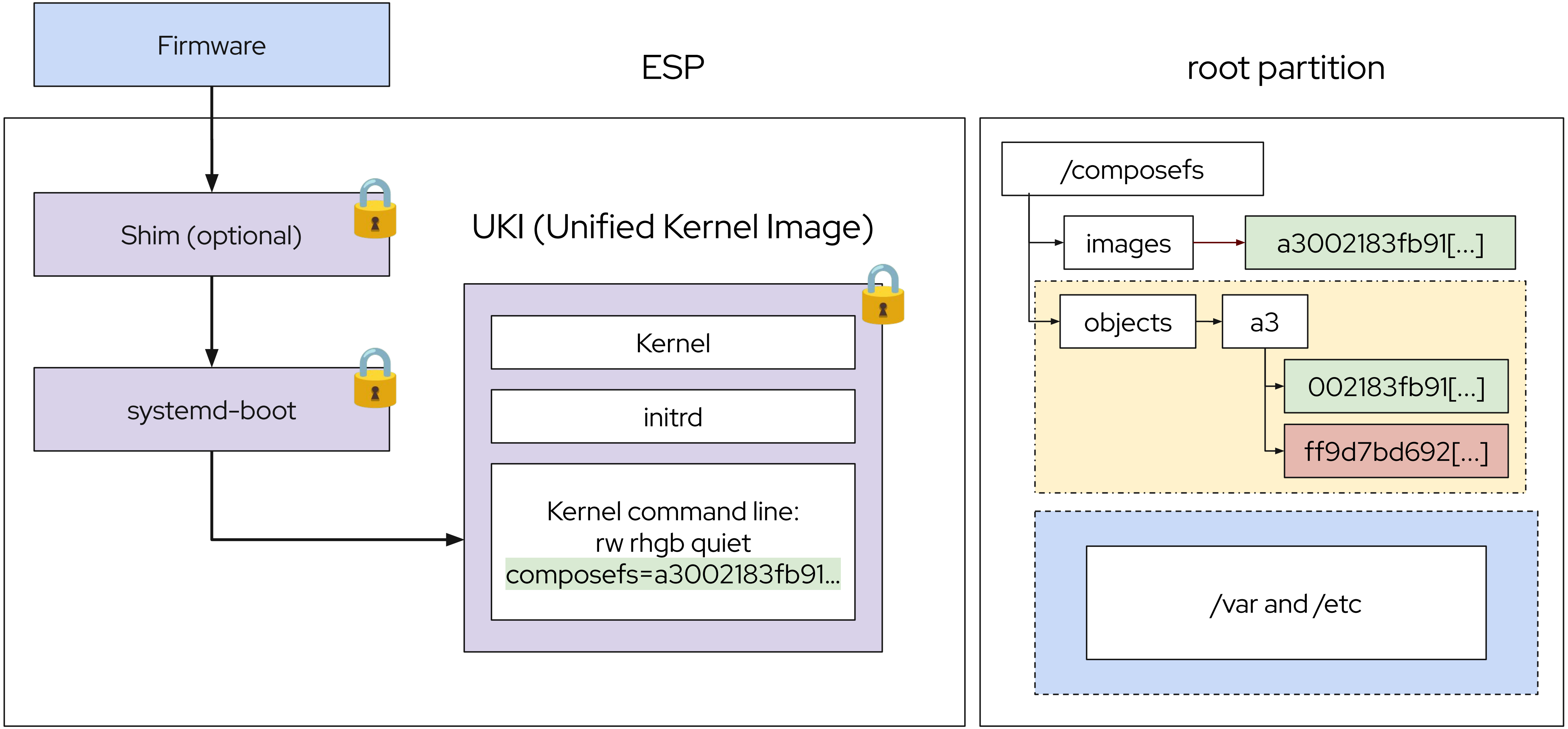
002183fb91[...]

ff9d7bd692[...]

/var and /etc

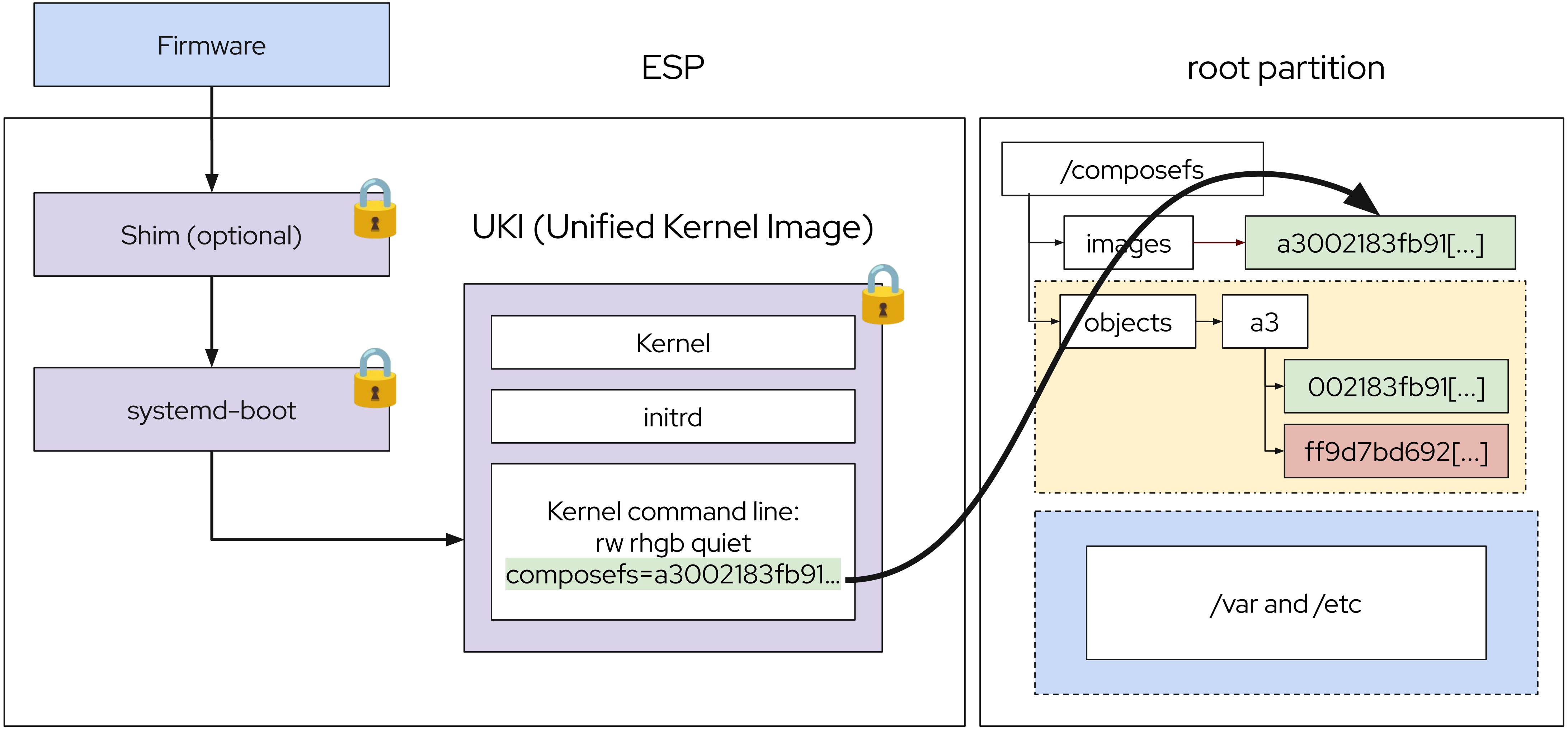
🔒 Signed for Secure Boot

Building a fully verified boot chain



🔒 Signed for Secure Boot

Building a fully verified boot chain



🔒 Signed for Secure Boot

Build your OS with a Containerfile

1. Take a regular container image
2. Install systemd, kernel, composefs and SELinux policy
3. Compute rootfs digest
4. Build the UKI, injecting the digest in the command line
5. Sign and add the UKI to the container image

composefs-rs project

https://github.com/containers/composefs-rs

Support for converting layered container images into root filesystems

Includes SELinux relabelling

Includes boot resources handling (via Boot Loader Specification)

Install updates on running systems

Still work in progress. Please come help us! (see future plans)

Demo!

Future plans

composefs-rs project mostly distribution agnostic

Plans to integrate with bootc (bootable containers) project

Use it for application container images and Flatpaks

Composefs and bootc accepted as Cloud Native Computing Foundation (CNCF) sandbox projects

- https://github.com/containers/composefs-rs
- https://github.com/containers/composefs
- https://github.com/containers/bootc

Thank you

Great, but dm-verity can do that too...

composefs brings many benefits:

- Simple partitioning layout: single filesystem and partition
- Arbitrary number of deploys / rollbacks
- No need for fixed partitioning (A/B, A/B/C/D? and how big?)
- No duplicated disk usage
- Can share file content (and memory usage) with containers as well

The catch

Kernel filesystem code is not robust enough against exploits

dm-verity protects the kernel from itself

In practice, most systems will store the content in a LUKS partition, using dm-crypt and optionally dm-integrity

Complete factory reset is not as easy as deleting a state partition:

- Must re-create the partition and re-install system content

Build your OS with a Containerfile

```
FROM fedora:rawhide AS base
COPY extra /
RUN dnf install -y selinux-policy-targeted systemd composefs

FROM base AS kernel
RUN --mount=type=bind,from=base,target=/mnt/base \
   COMPOSEFS_FSVERITY="$(cfsctl --repo /tmp/sysroot create-image /mnt/base)" \
   && echo "composefs=${COMPOSEFS_FSVERITY} rw" > /etc/kernel/cmdline

RUN --mount=type=secret,id=key --mount=type=secret,id=cert \
   dnf install -y kernel systemd-boot-unsigned systemd-ukify sbsigntools

FROM base AS bootable
COPY --from=kernel /boot /composefs-meta/boot
RUN rm -rf /composefs-meta
```