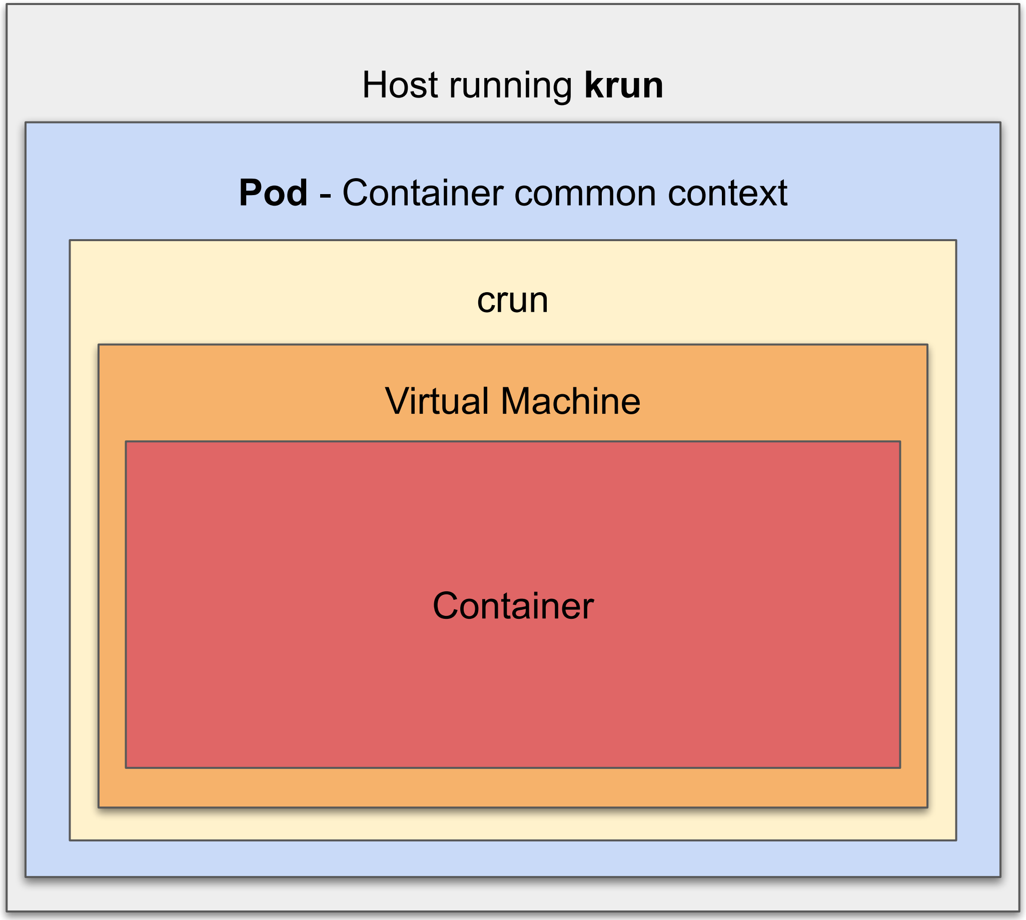
Attestable Containers with keybroker

Tyler Fanelli, Red Hat

Background: `krun` Container Runtime

- Extension of `crun` runtime with added virtualization capabilities.
- Allows containers to run inside micro-VMs, a machine type optimized for boot time and memory footprint.
  - Much smaller footprint than standard VMs, as `krun` only provides a “thin layer of virtualization” wrapping a container’s workload with a minimal amount of devices.
- Added SEV-SNP support (with attestation).



keybroker

- Simplified attestation server that re-uses the KBS protocol found in the Trustee (formerly CoCo KBS) server.
    - Experimenting with keybroker usage of the “attestation-service” (backend) of Trustee service.
- Follows RATS
- Modifications around registration and handling of registration values needed for running “confidential container images” (to be explained).

Container Images for Confidential Computing

- Users want confidence that their workloads are running confidentially.
- How can we build container images specifically with confidential computing in mind?
  - Is there a way we can “force” attestation of a container?
- Can we build container images such that attestation is required for running the container?
- Can we leverage existing container image builders to create images for confidential computing?
- How can we make the process of building “confidential container images” as simple as possible?

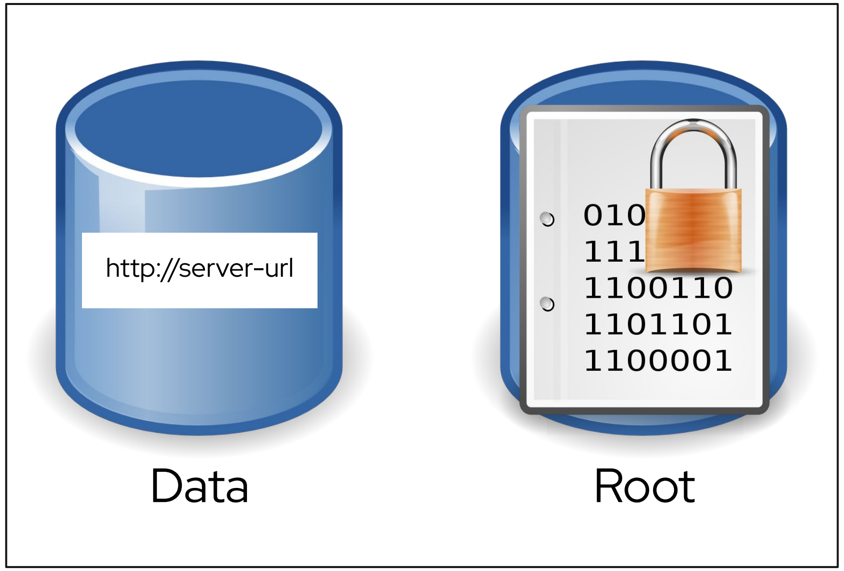
Component: buildah

- Tool to build OCI container images.
- Two new features:
  - `buildah build`, the main command to build OCI images, recently introduced the `--cw` (confidential workload) flag to build OCI images with **`krun`** VM attestation in mind
  - `buildah mkcw` command, which takes an existing container image and converts it into a “confidential workload” image.
- Both commands:
  - Build “confidential workload” images.
  - Register build information with an attestation server.

Red Hat

buildah Confidential Workload Images

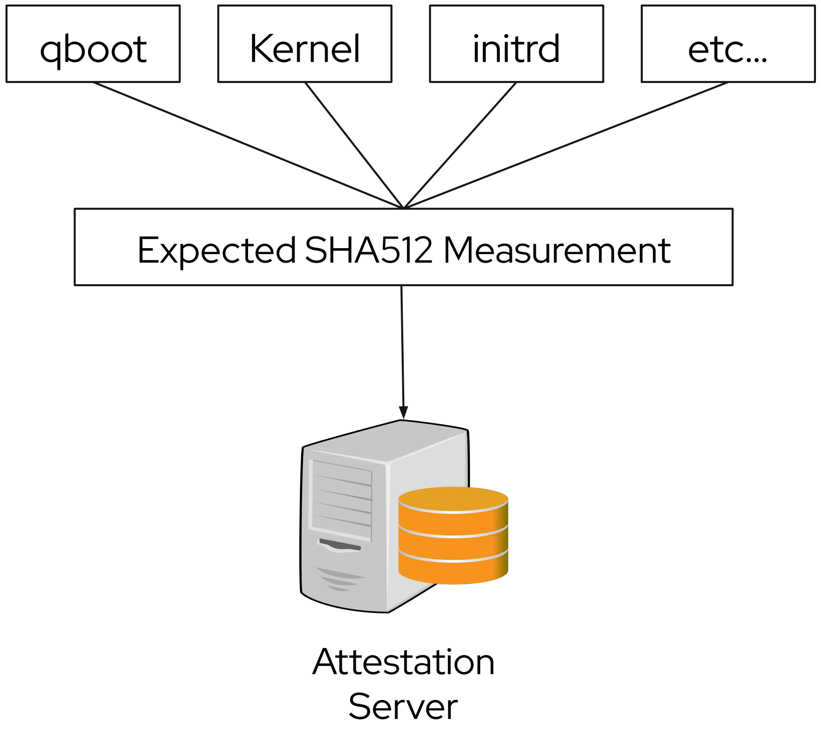
- Expected to be run with `krun` runtime.
- Contain two main disks expected by `krun`.
  - Root disk
    - Contains application code/data.
    - LUKS encrypted, with key stored in attestation server (more later).
  - Data disk
    - Contains data needed for attestation, such as the URL of the remote attestation server needing to communicate with.
- Since application (encrypted in root disk) cannot be ran until unlocked, attestation is required for all workloads.



buildah `--cw` Image

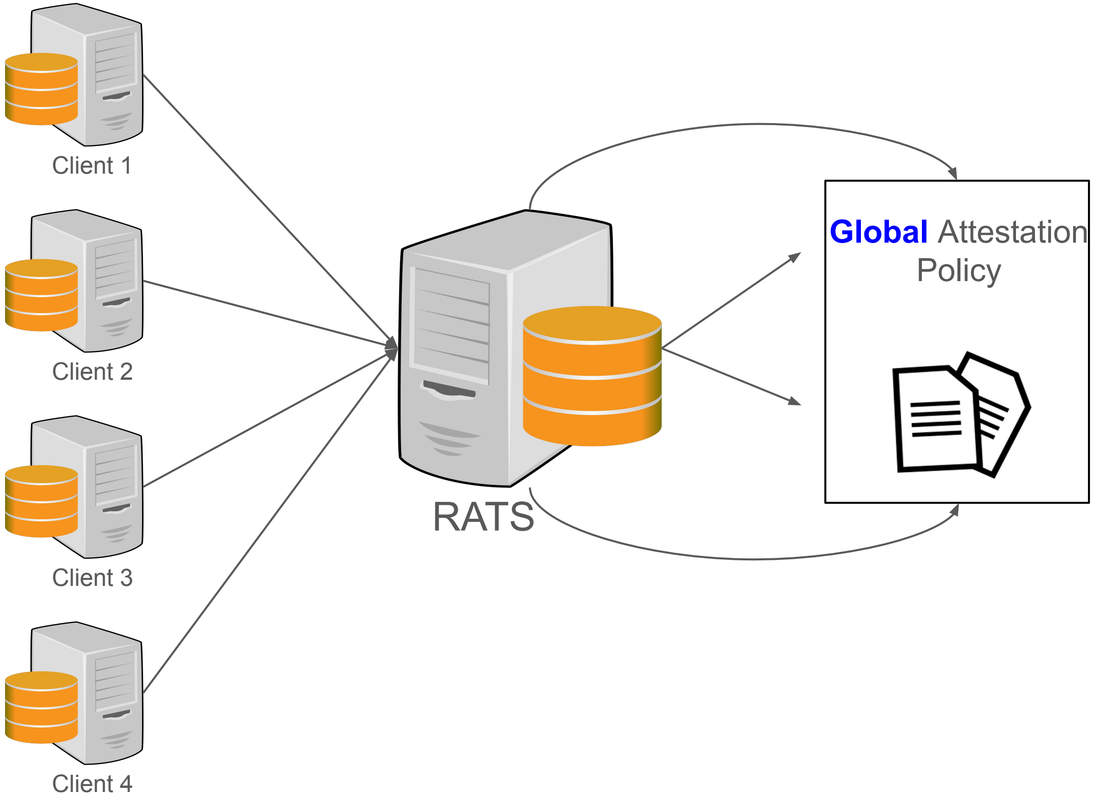
buildah Confidential Workload Registration

- Buildah measures each VM component that will be used to run workload and computes a hash.
    - qboot, kernel, initrd, etc..
- Hash is registered with the attestation server as the **expected launch measurement**.
- URL of attestation server is inserted into image data disk.
- NOTE: All registration of reference values, other data needed for attestation done at build time of **container image**.
    - Any user of image must attest with attestation server that the image was **built to attest with**.
    - Whoever runs the image **must** attest it.



Traditional RATS Server Approaches

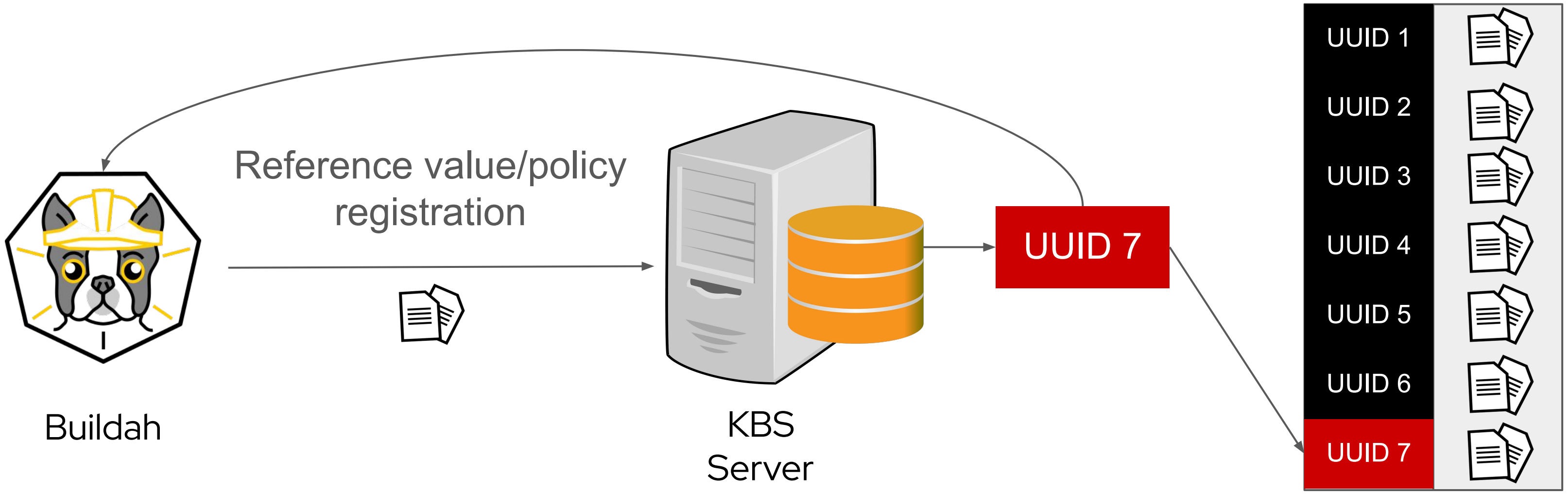
- **Every** client must attest according to a global attestation policy.
- Evidence must be presented in a form that attestation policy can interpret.
- Reference values must be submitted with policy.



Problems?

- Not flexible, all reference values/policies must be supplied in one global attestation policy.
- Doesn't fit with the buildah confidential container image model.
  - No assumptions made about the workload (reference values, TEE arch, policy) to be attested.
  - Users cannot submit their own policies when building confidential container images.
- All reference values and policies are workload-specific and aren't generated until build time.
- Would rather have a registration interface that is client-specific, where users could register their own reference values/policies and attest using them.
  - More flexible.

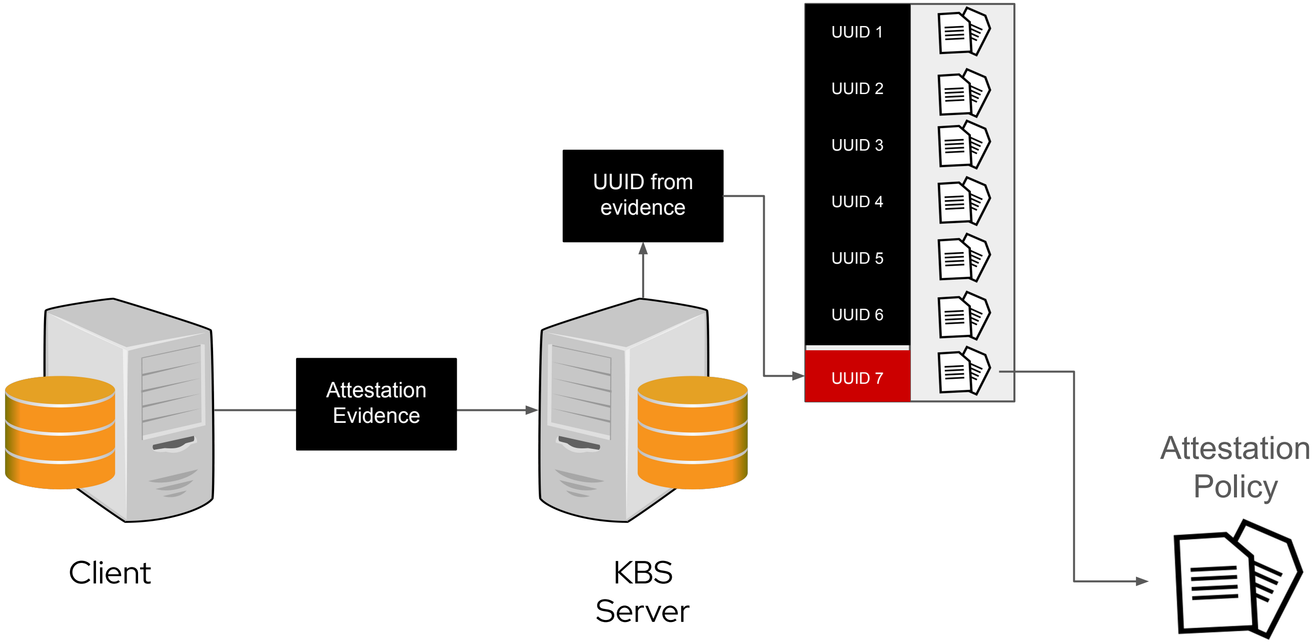
keybroker Registration



- keybroker maintains a registration endpoint, handling POST requests to register clients for later attestation.
- User registers attestation reference values and policy.
- Server generates a UUID and stores the policy in a map with the UUID as a key.
- UUID is returned to buildah. Buildah inserts ID into container image's data disk for attestation.

keybroker Attestation

- Guest attests, presenting the UUID that buildah received at registration.
- Server parses this ID from evidence.
- Server uses this ID to index into the reference value map.
- Server retrieves the policy corresponding to the client and uses reference values/policy to attest.

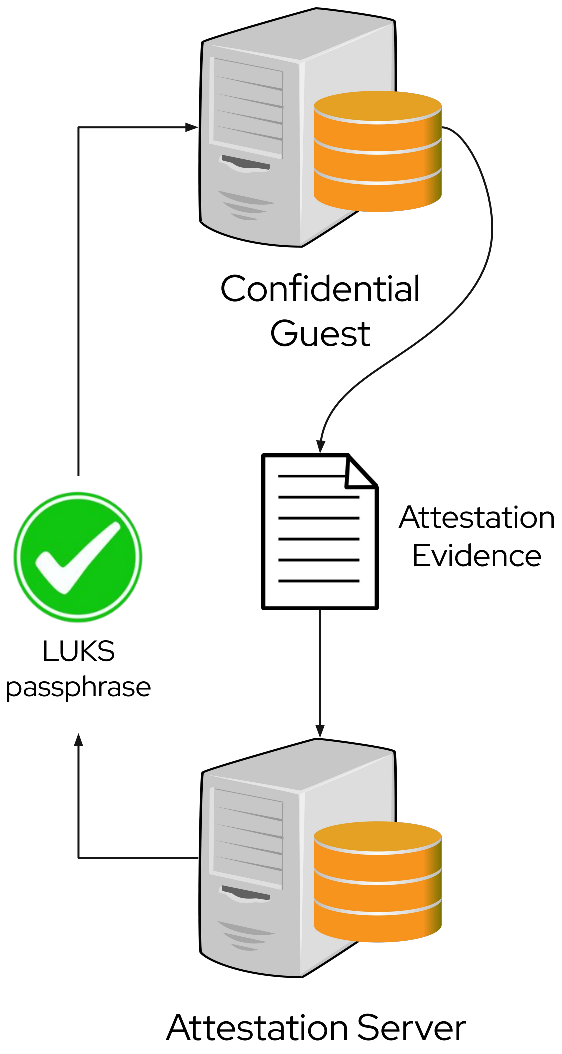


Attesting and Running Confidential Container Images

- We now have container images that must be attested to run.
- How can we use an existing container runtime to do this attestation, unlock the root disk, and run the container's contents virtualized?
- Can we hide all of this from the user, such that it's completely transparent?
- Can we hide the fact the container is running virtualized from the user?

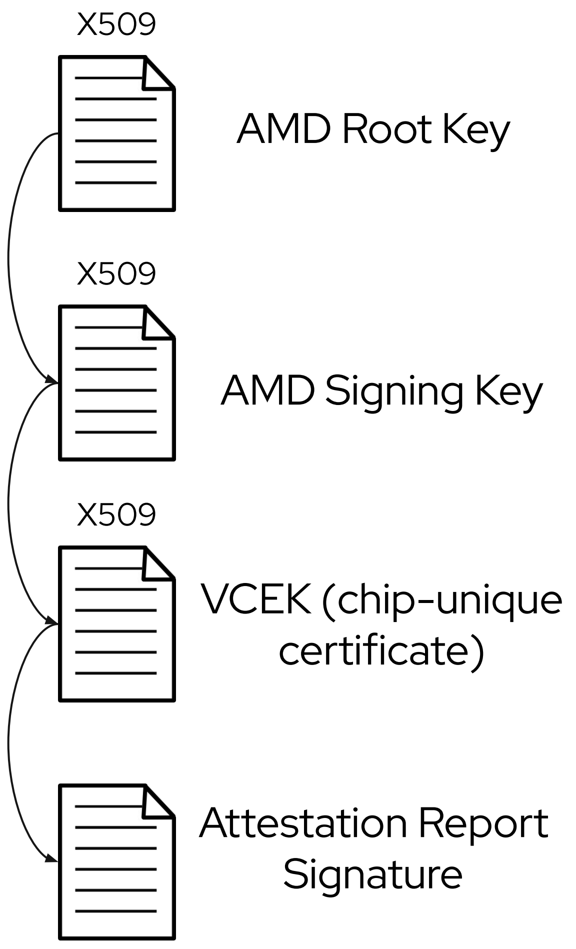
krun Attestation

- Each TEE architecture allows guests to request an attestation report from the secure processor.
  - Launch measurements, identification information, etc…
- krun fetches and sends an attestation report to the attestation server URL (read from data disk).
- Attestation server completes attestation and reports results.
- If successful, server sends VM the LUKS passphrase to unlock the root disk and begin running the application.



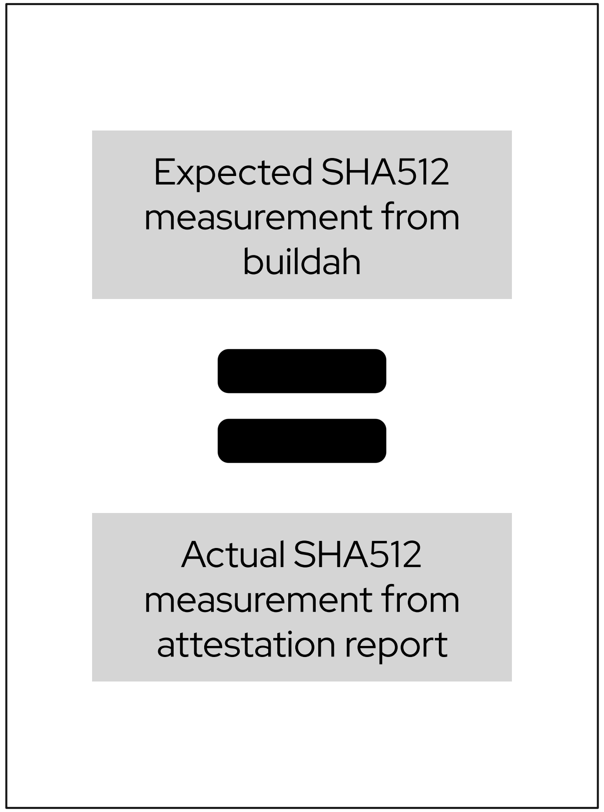
Attestation Server: Hardware Validation

- Verify that we're running on authentic TEE hardware from chip supplier.
- Each attestation report contains a signature that can be traced back to the chip supplier's root of trust.
  - SEV-SNP given as example. Certificate chain traced back to AMD root of trust.
- Cryptographically proves that the attestation report is from an authentic TEE processor.
- Host cannot lie about running confidentially.



Attestation Server: Software Validation

- Recall that `buildah build --cw` previously registered an expected launch measurement with the attestation server.
- The (now authenticated) attestation report contains a hash of all components encrypted by the secure processor (i.e. the **actual launch measurement**)
- Must compare the expected launch measurement with the actual launch measurement.
- If not equal, either:
  - Not all VM pages were encrypted by secure processor.
  - Extra pages were mapped into VM guest memory.
- Either option may jeopardize the confidentiality of the workload, so attestation fails if expected and actual launch measurements do not match.

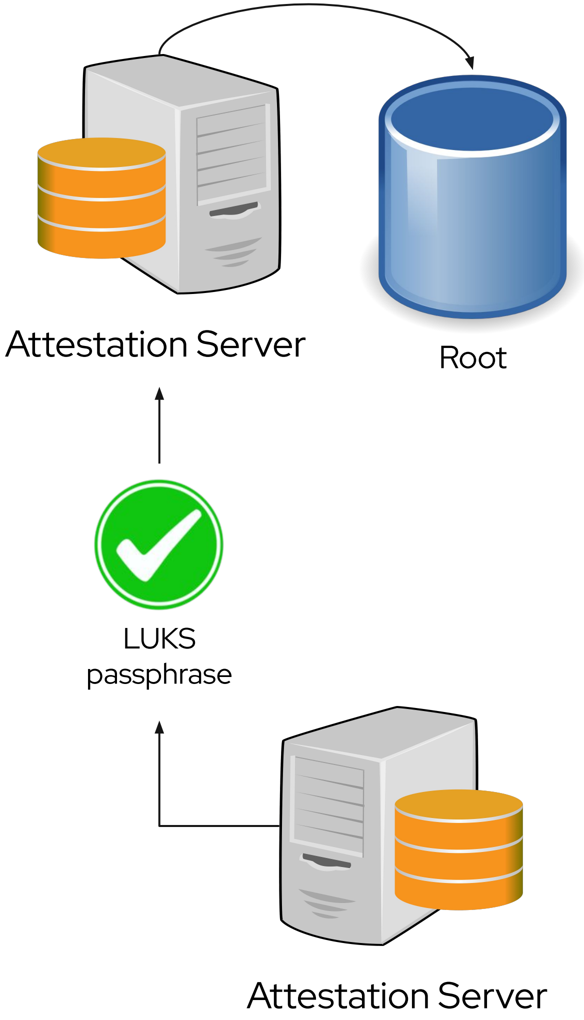


Expected SHA512 measurement from buildah

Actual SHA512 measurement from attestation report

krun VM: Successful Attestation

- On successful attestation, attestation server sends LUKS passphrase to VM.
- VM uses LUKS passphrase to unlock root disk and begin running application.
- Cryptographically proven that our workload (and only our workload) is running confidentially on the host system.

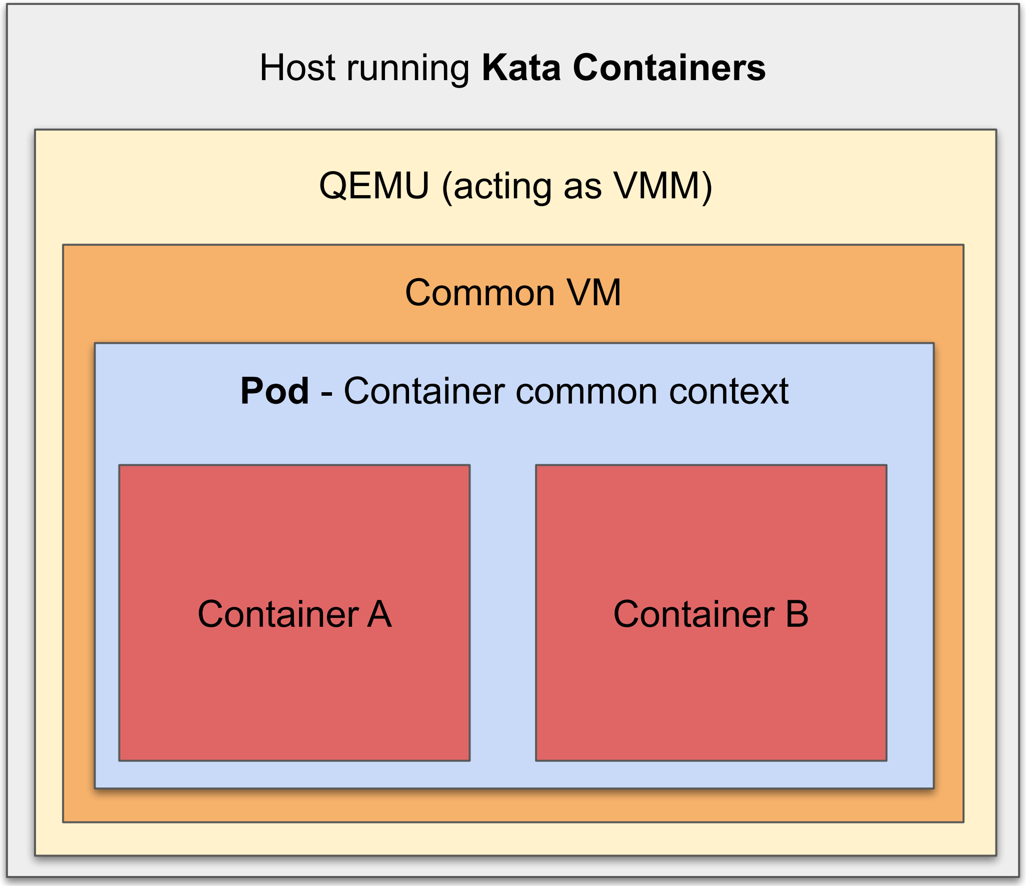


SEV-SNP Demo

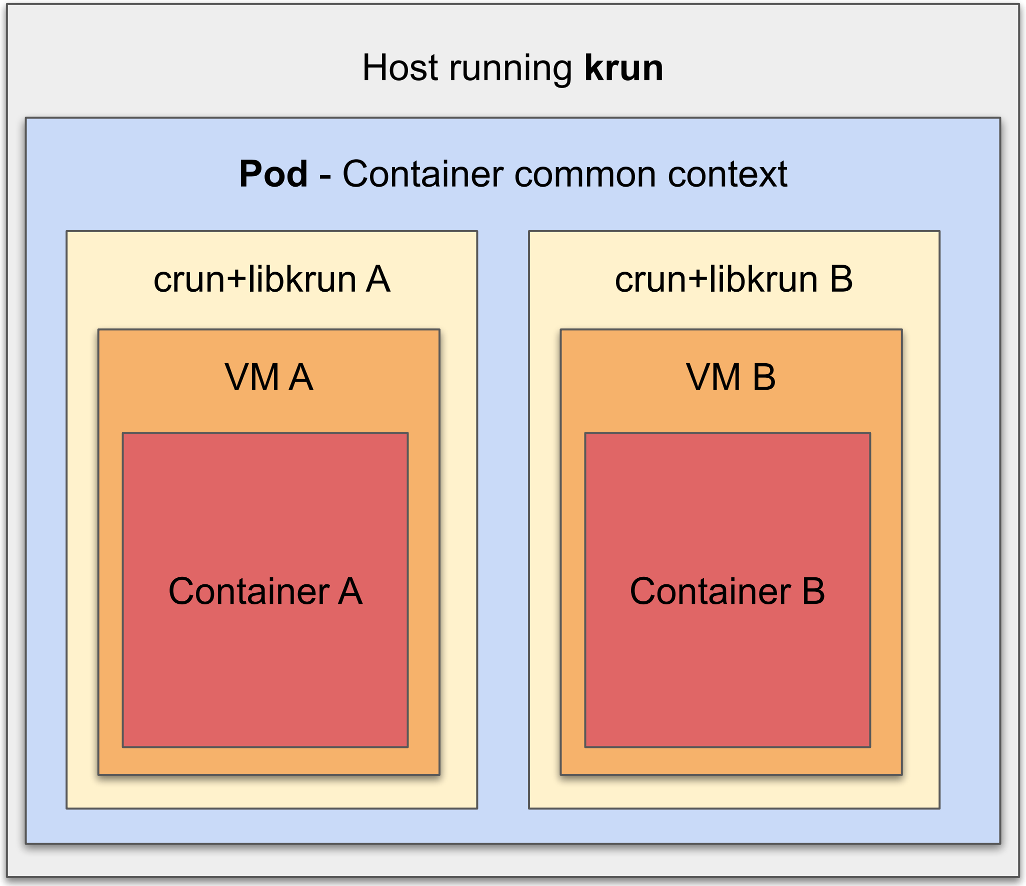
https://asciinema.org/a/653719

Questions?

Aside: krun Runtime vs. Kata Containers



Host running **Kata Containers**
QEMU (acting as VMM)
Common VM
Pod - Container common context
Container A
Container B



krun Runtime vs. Kata Containers

- Kata Containers advantages
  - Supports most container workloads.
  - Built on mature components (QEMU).
  - Private environment between containers in the same Pod.

- krun advantages
  - Simpler, fewer moving pieces.
  - Doesn’t need to grow/shrink the VM.
  - Smaller attack surface.
  - Lower per-VM memory footprint.

Thanks!