# Measurement and Attestation Schemes for Container Sandboxes

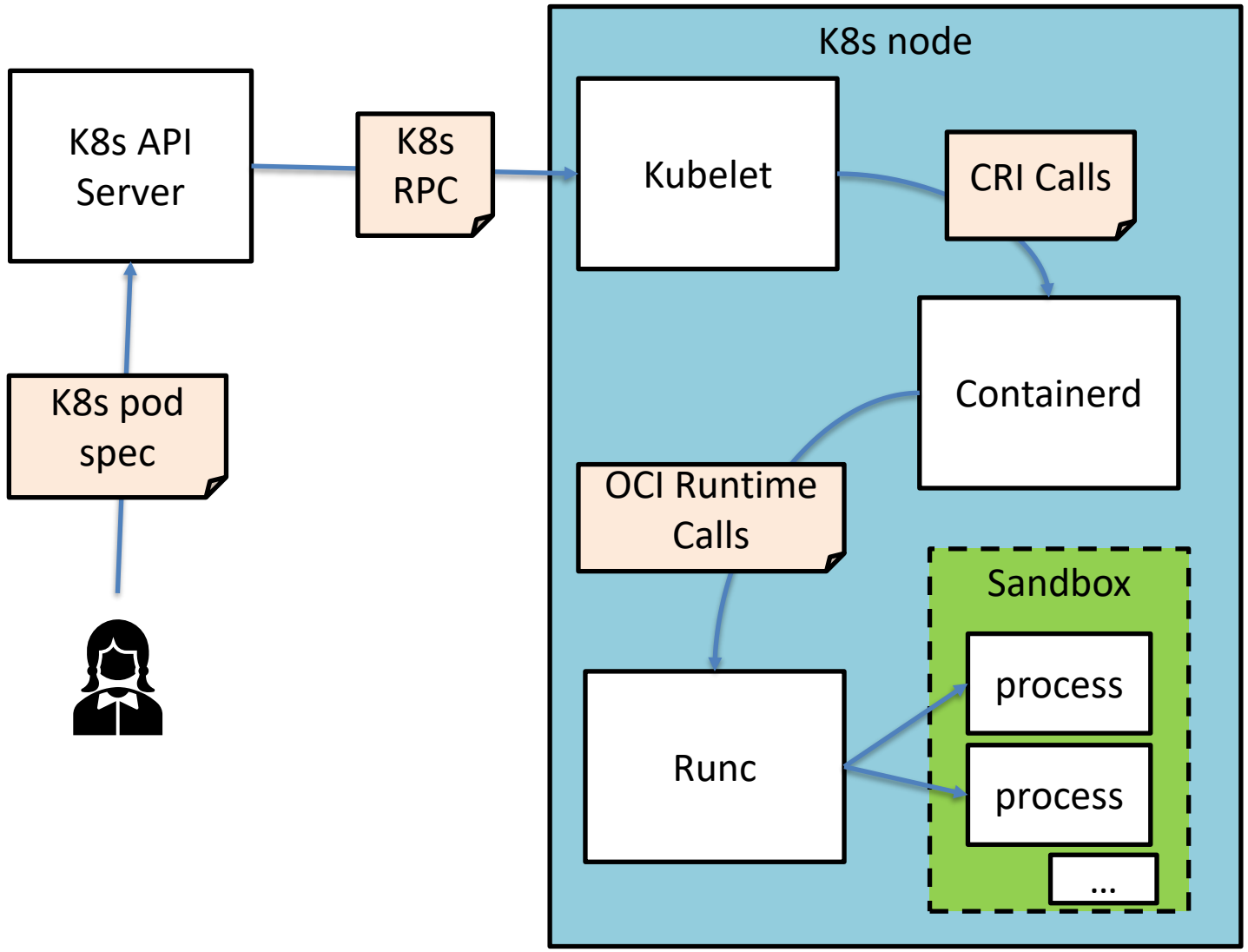
Magnus Kulke, swe @Azure Core Linux

FOSDEM 25, Attestation Devroom

# Context

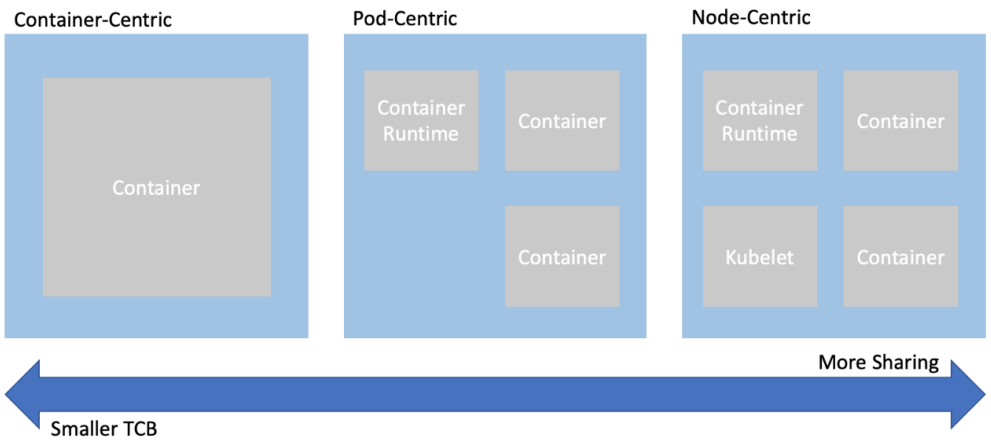
- Confidential Containers (CoCo)
  - CNCF project
  - vendor neutral
  - Facilitate confidential computing in the container ecosystem
- Confidential Computing is (mostly) a VM technology
  - Containers usually do not run in VMs
  - Kata Containers adds virtualization as isolation layer

# What is a typical container launch?

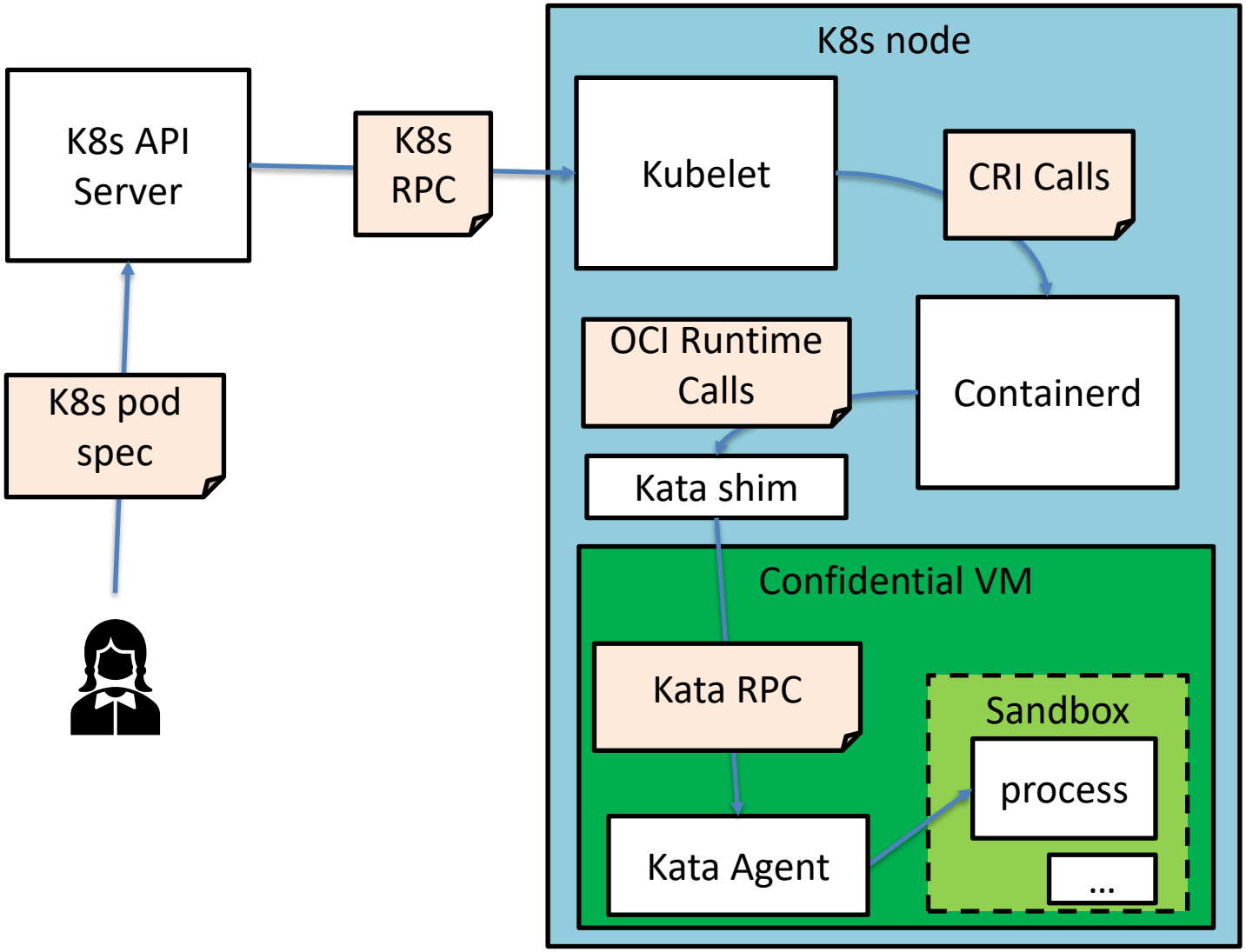


# Sandbox?

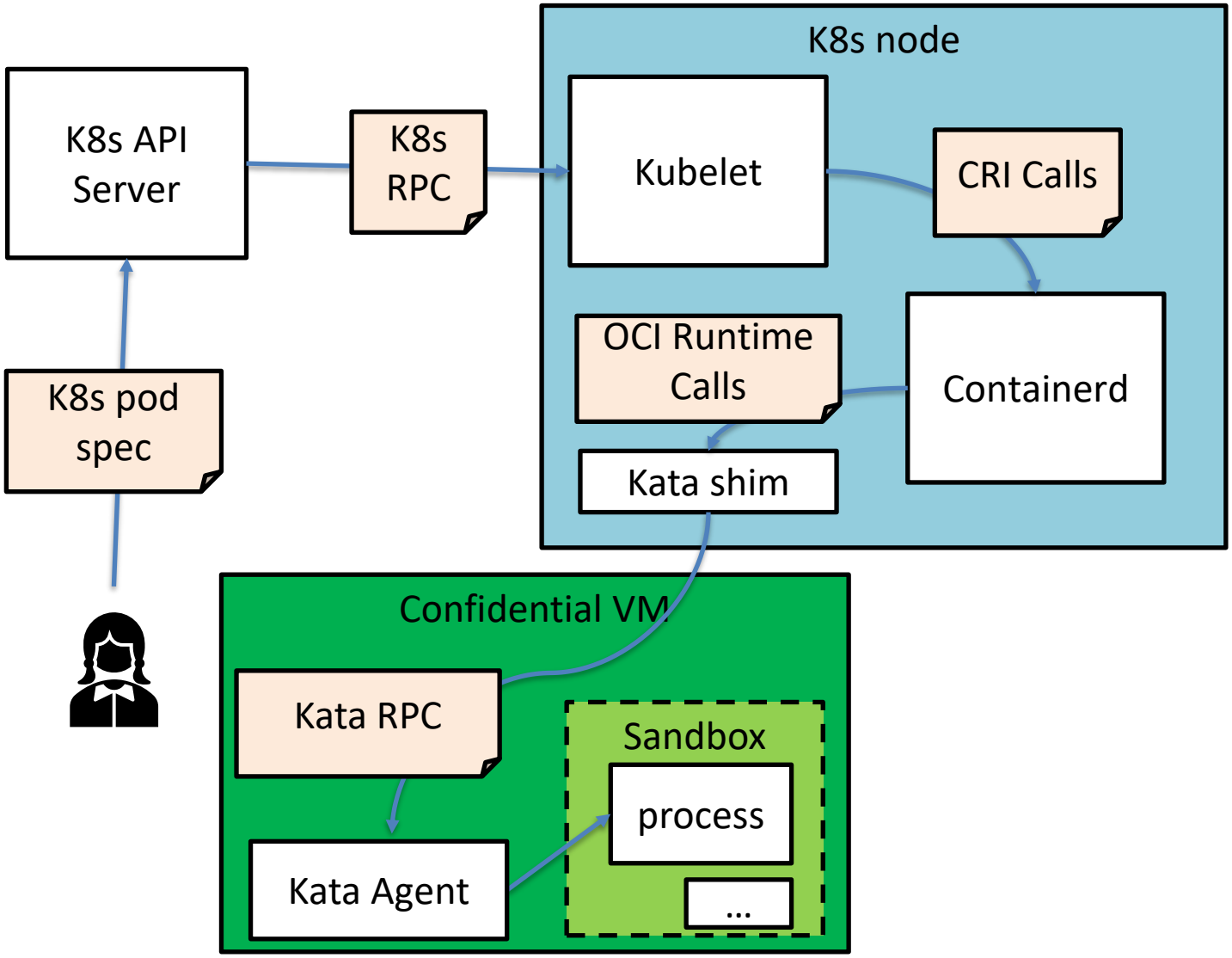
- Pod: Kubernetes deployment „atom“.
- Set of collocated processes (containers) that share namespaces and resources
- Good abstraction to introduce confidentiality boundaries



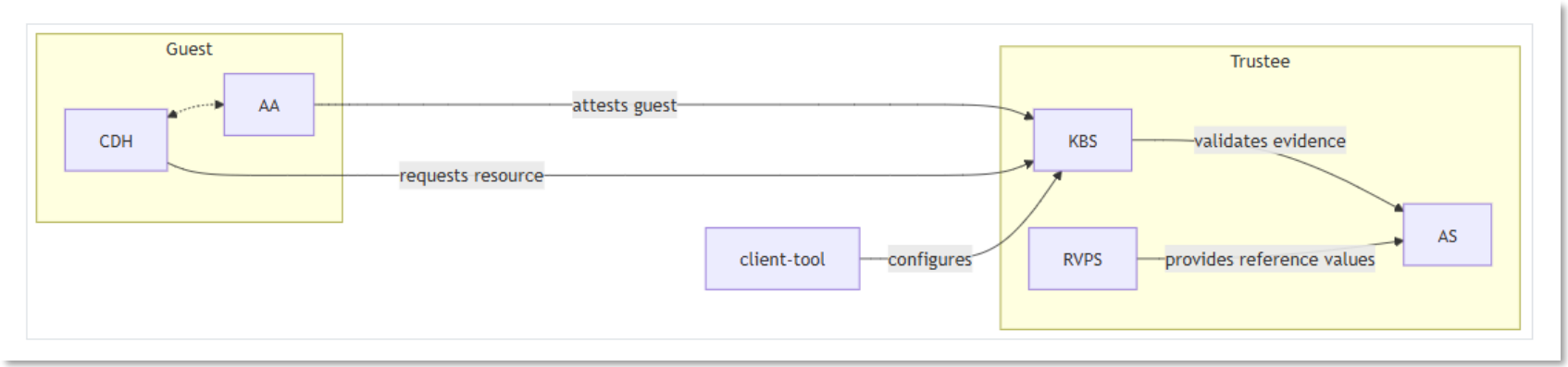
# Confidential container launch



# … or with a remote CVM



# Attestation Architecture

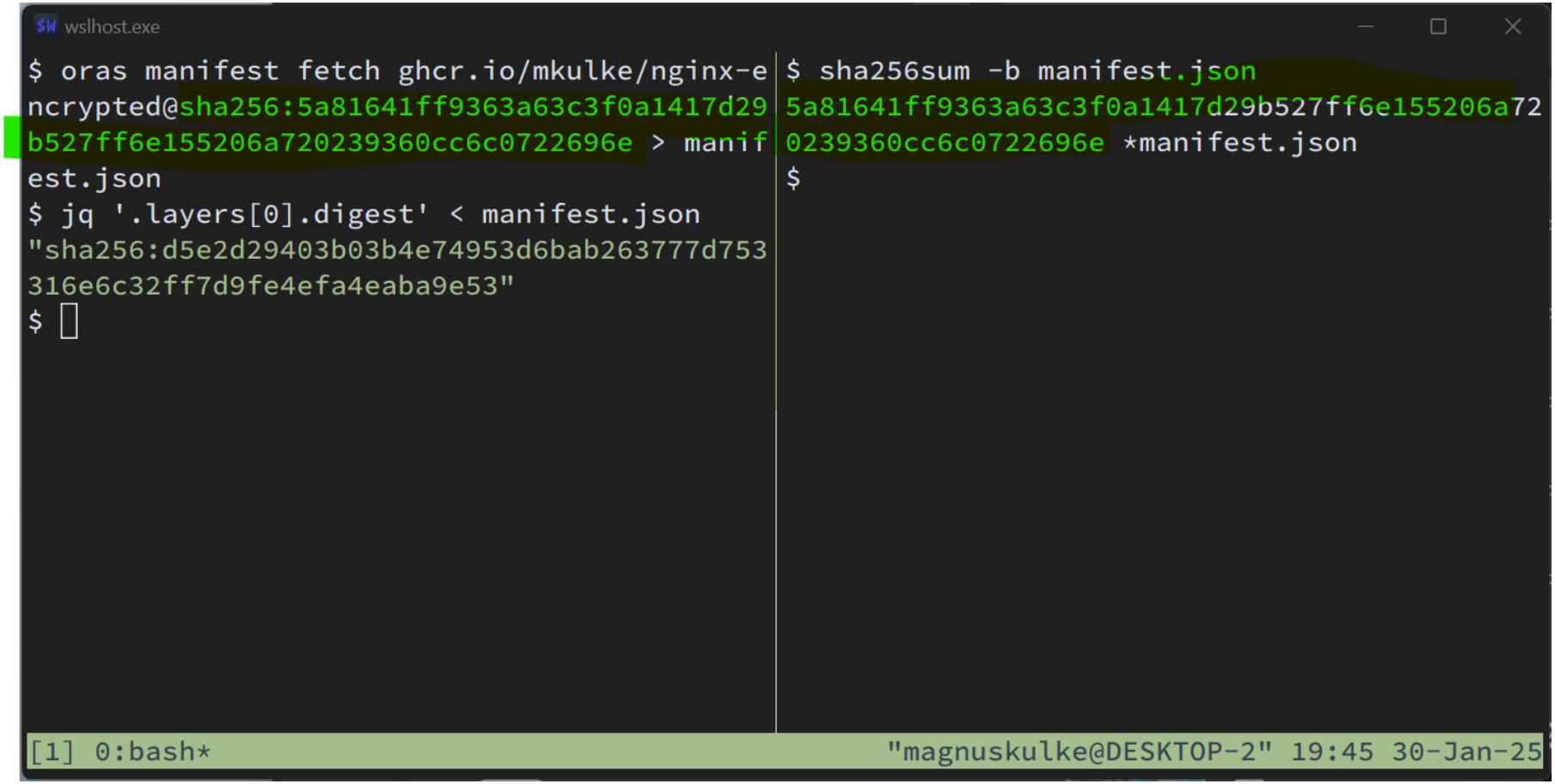


# Static and dynamic components

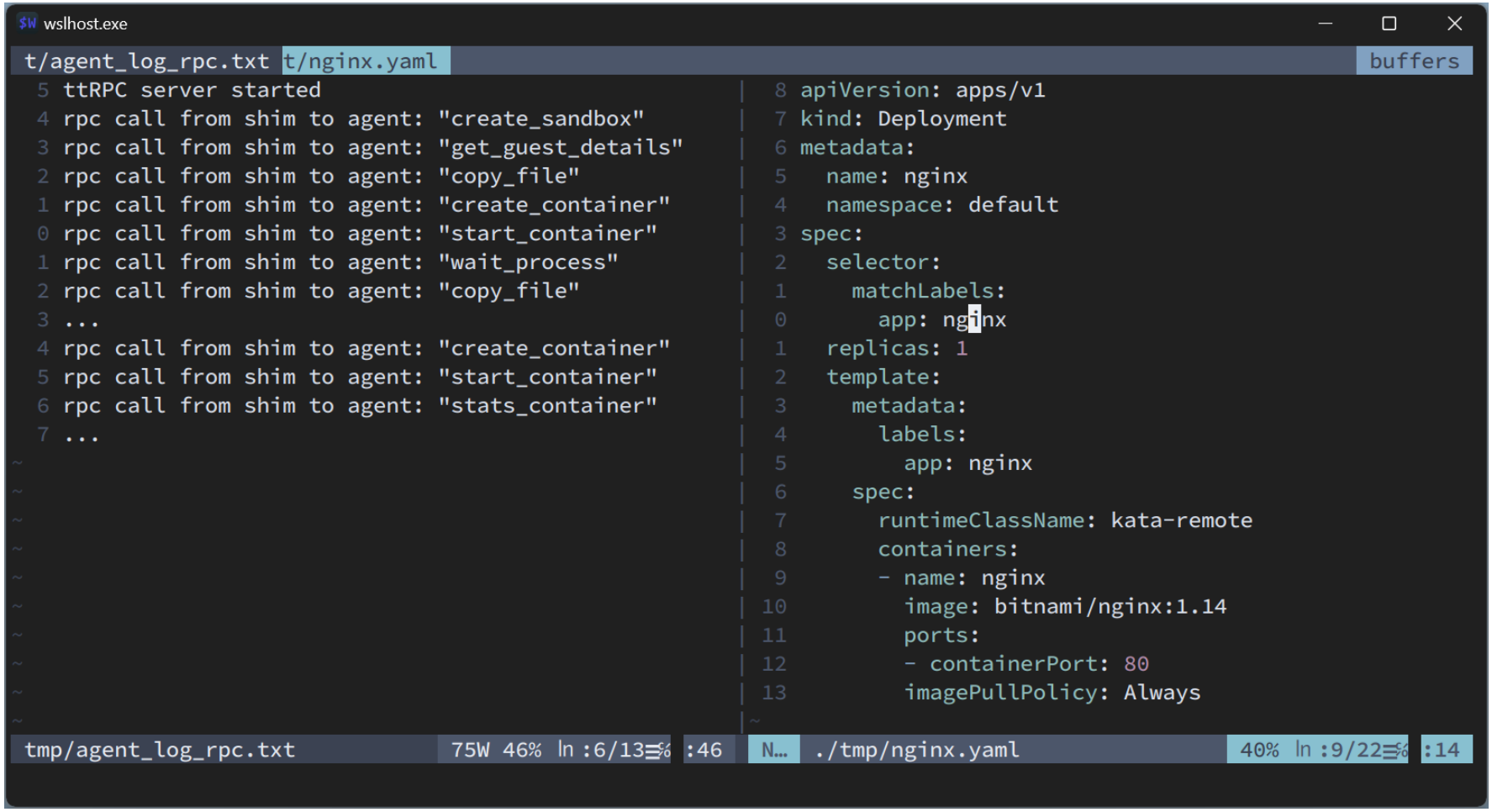
?

Well covered: UKIs, dm-verity, systemd…

# OCI images are content addressable



# Sandbox with imperative control



# Attesting a container environment

Objective:

Ensure that only intended operations are executed within the sandbox (before releasing a secret)

Requires:

A comprehensive measurement of the “container workload”

# Challenges in dynamic environments

- Dynamic Nature of Pods:
  - Pods can have containers created, deleted, or updated imperatively.
  - Dynamisms make it challenging to guarantee integrity.
- Kubernetes Control Plane:
  - Can and will adjust a user's pod spec
  - Examples: env variables, admission controllers

# Options

- Lock k8s control plane:
  - Allow only “trusted” (predictable) operations
  - K8s api surface is huge, increasing constantly
  - Requires lots of glue code and ceremony
- Effort underway: “split-api”

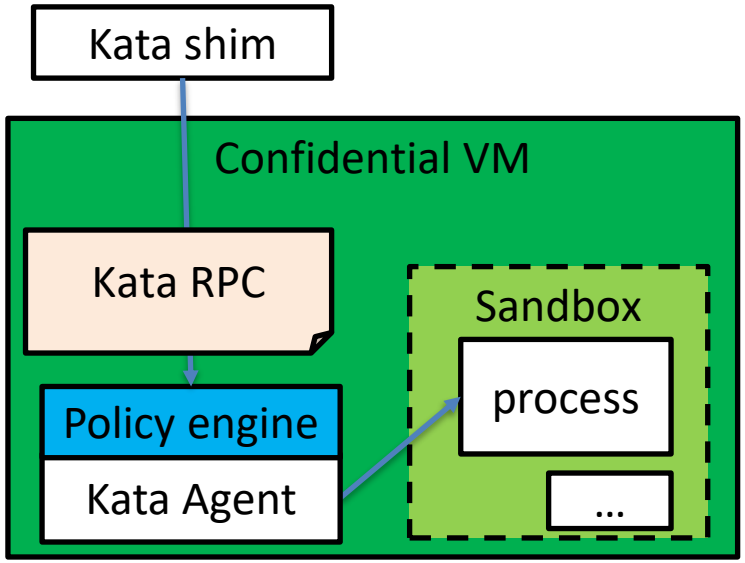
# Options

- Keep a log
  - (Somewhat) like linux' IMA
  - Record Kata RPC + payloads into replayable log
  - Not all TEEs provide registers that can be extended at runtime
  - Some payloads are not predictable, b/c controlled by the env
  - Verification is not trivial

# Options

- Policy in the TEE
  - Describe invariants (image digest)
  - Allow “acceptable” dynamism (env: SERVICE_*)
  - Reject Kata RPCs by default
  - cherrypick what’s required
- Currently implemented in Kata-Agent
  - Engine based on Rego (popular in container-land)
  - genpolicy tool to generate policy from a pod spec

# Plugging policy eval into the workflow



# Example Policy

```
package agent_policy

import future.keywords.in
import future.keywords.if
import future.keywords.every

default CopyFileRequest := true
default DestroySandboxRequest := tru
default CreateSandboxRequest := true
default GuestDetailsRequest := true
default ReadStreamRequest := true
default RemoveContainerRequest := tr
default SignalProcessRequest := true
default StartContainerRequest := tru
default StatsContainerRequest := tru
default WaitProcessRequest := true

default CreateContainerRequest := false
default ExecProcessRequest := false

CreateContainerRequest if {
  every storage in input.storages {
        some allowed_image in policy_data.allowed_images
        storage.source == allowed_image
    }
}
```

```
ExecProcessRequest if {
    input_command = concat(" ", input.process.Args)
  some allowed_command in policy_data.allowed_commands
  input_command == allowed_command
}

policy_data := {
  "allowed_commands": [
    "whoami",
    "false",
    "curl -s http://127.0.0.1:8006/aa/token?token_type=kbs",
  ],
  "allowed_images": [
    "pause",
    "docker.io/library/nginx@sha256:e56797eab4a5300158cc015296229e1
  ],
}
```

# How to provide a policy to the TEE?

- Policy is specific per workload
- CVM images are generic
- Provide it as measured configuration at launch
- Link it to the TEE HW evidence
  - Put hash in HOSTDATA (SEV-SNP), MRCONFIGID (TDX), part of signed HW evidence (verify in TEE)
  - Extend runtime registers (vTPM)

# Init-Data Specification

- Measured configuration for CoCo
- TOML dict of path/file content
- Currently being implemented
- Available for some TEEs
- Embed into Pod spec as annotation

```
vim init-data.toml
INIT_DATA_B64="$(cat "init-data.toml" | base64 -w0)"
cat nginx-cc.yaml | jq \
  --arg initdata "$INIT_DATA_B64" \
  '.spec.template.metadata.annotations = { "io.katacontainers.config.r
  | kubecl apply -f -
```

# Initdata Example

```
algorithm = "sha256"
version = "0.1.0"

[data]
"aa.toml" = '''
[token_configs]
[token_configs.kbs]
url = 'http://my-as:8080'
cert = """
-----BEGIN CERTIFICATE-----
MIIDEjCCAfqgAwIBAgIUZYcKIJD3QB/LG0FnacDyR1KhoikwDQYJKoZIhvcNAQEL
...
4La0LJGguzEN7y9P59TS4b3E9xFyTg==
-----END CERTIFICATE-----
"""
'''

"cdh.toml"  = '''
socket = 'unix:///run/confidential-containers/cdh.sock'
credentials = []
...
'''

"policy.rego" = '''
package agent_policy
...
```

# Challenges

- Policy is stateless, declarative
  - Kata RPC is imperative
  - What about more complex orchestration?
    - launch container x first (init container) then container y

- Ongoing effort: stateful policies

# Challenges

- Practical problems
  - Size of policies?
    - Policies can be quite large
    - Pod annotation has limits
    - Compression, splitting, bundling a library
  - User experience is subpar
    - Rego is modelled after Datalog
    - Unusual paradigms
    - Not trivial to write large policies

# Challenges

- Conceptual problems, maybe?
  - Have to track kata's RPC interface closely
    - New exploit vectors can be introduced inadvertently
    - Kata is not just for CoCo use case
    - Need to keep tabs on API changes in semantics and implementation

# Challenges

- Runtime measurements
  - (very) long running workloads in TEEs
    - Examples: LLM inference, training tasks
    - Continuous measurement to catch drift
  - Not all TEEs have PCRs/RTMRs
  - Can be retrofitted via privilege levels + paravisor/SVSMs.
- “Composite” TEEs
  - Confidential GPUs + Confidential CPUs
  - Potentially more, e.g. accelerated NICs
  - Attest individually? Chained?

# Recap

- Attestation for container sandboxes is tricky due to inherent dynamic nature.
- “Offloading” verification to a policy is a viable mitigation
- Few challenges remain, most seem manageable
- But policy is maybe not fully adequate

thx!

# References

- Confidential Containers
- Kata Containers - Open Source Container Runtime
- Policing a Sandbox | Microsoft Community Hub
- CoCo Initdata spec