AIRFLOW SUMMIT

marvik.

Beyond Containers:

Securely Orchestrating AI Agents with Strong Isolation in Airflow

Diego Zimmerman and Uriel Muñoz


marvik.
Built for the Now

marvik.

Marvik is a hands-on AI consulting firm **built for the now.**

We design and deploy AI solutions that scale with your business and keep you competitive in industries where speed and impact matter most.

Partnering with those shaping the future of AI:

ORACLE Google Cloud aws Microsoft

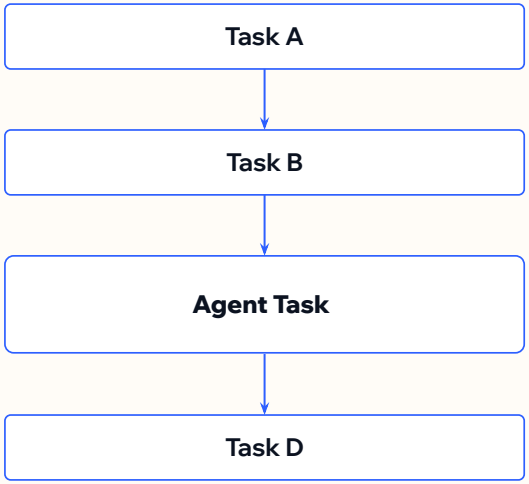
ANTHROP\C OpenAI NVIDIA IIElevenLabs

Trusted by **top clients**

Agenda

1. **The AI Workload Shift**
   What changes when your DAG tasks execute code generated at runtime instead of static code you reviewed.
2. **The Shared Kernel Risk**
   How Airflow's KubernetesExecutor runs task pods, and why sharing the host kernel creates security exposure.
3. **Four Isolation Primitives**
   A practical look at key workload isolation options you can plug into Kubernetes to contain untrusted execution.
4. **Trade-offs & Alternatives**
   Performance, setup, and complexity trade-offs, plus native Airflow operators if you prefer offloading compute isolation entirely.

What changes when Airflow runs AI agents?

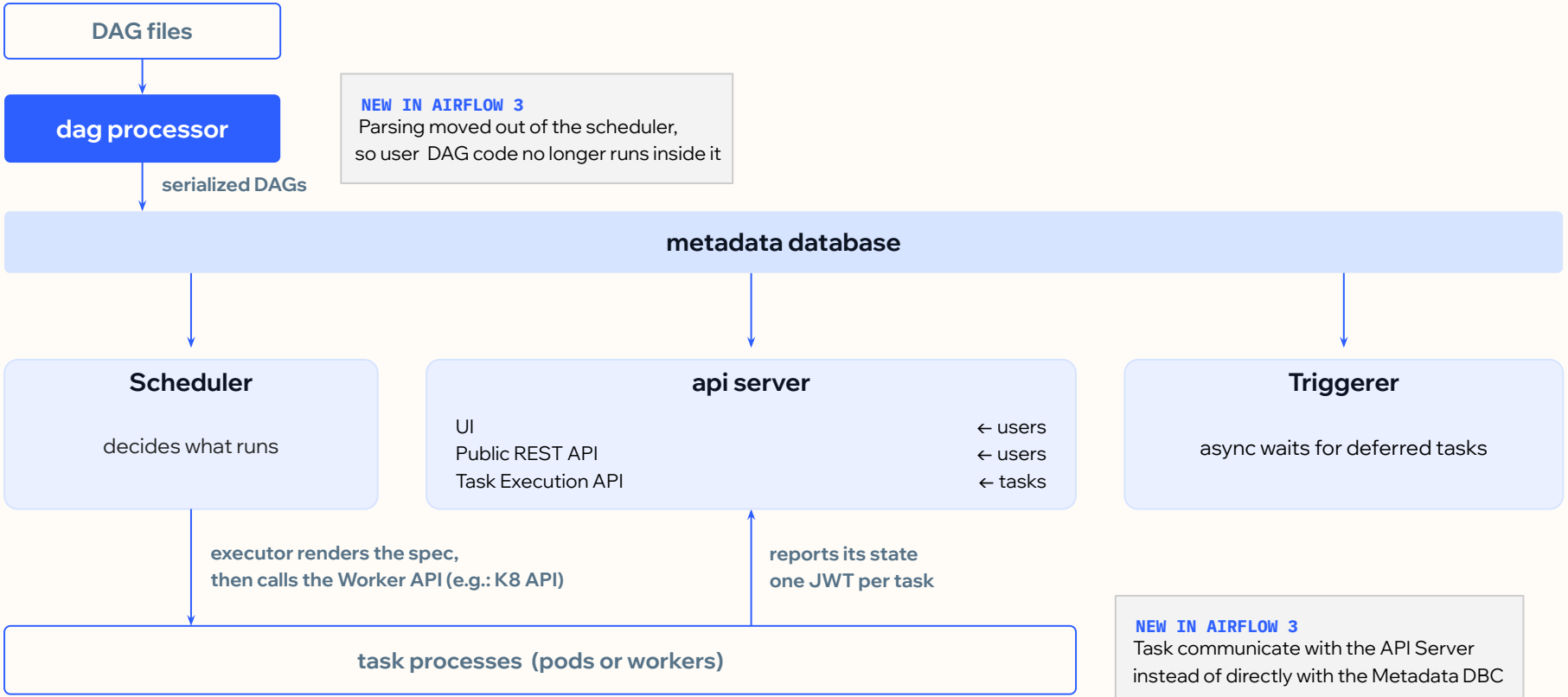


AI agents dynamically decide what code to generate, commands to execute, tools to call, and retry paths to follow at runtime.

This shifts the operational profile: we are no longer running static code we reviewed, but semi-autonomous code execution loops inside our cluster.

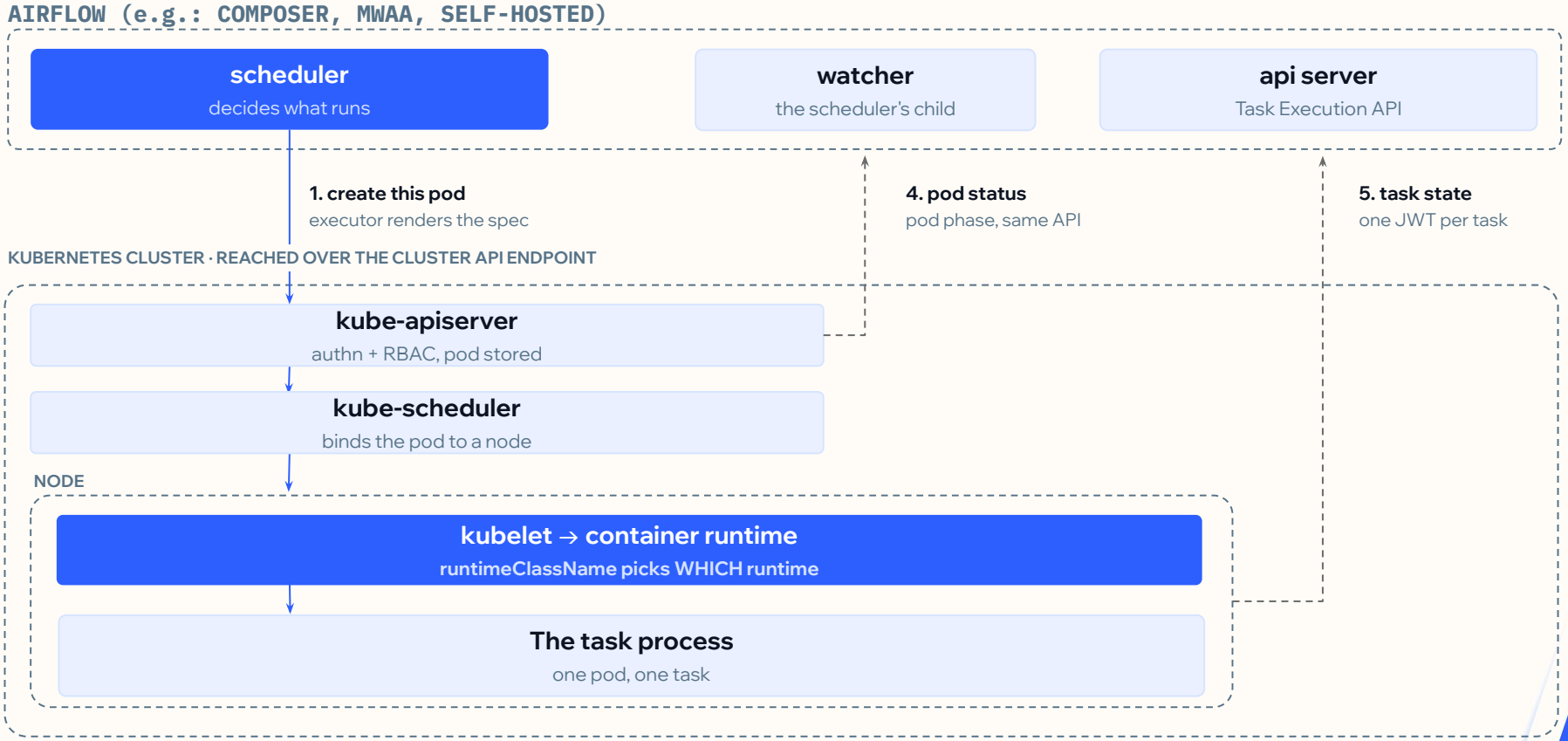
Core Question: How much do we trust the workload we are executing on shared hardware?

Airflow anatomy

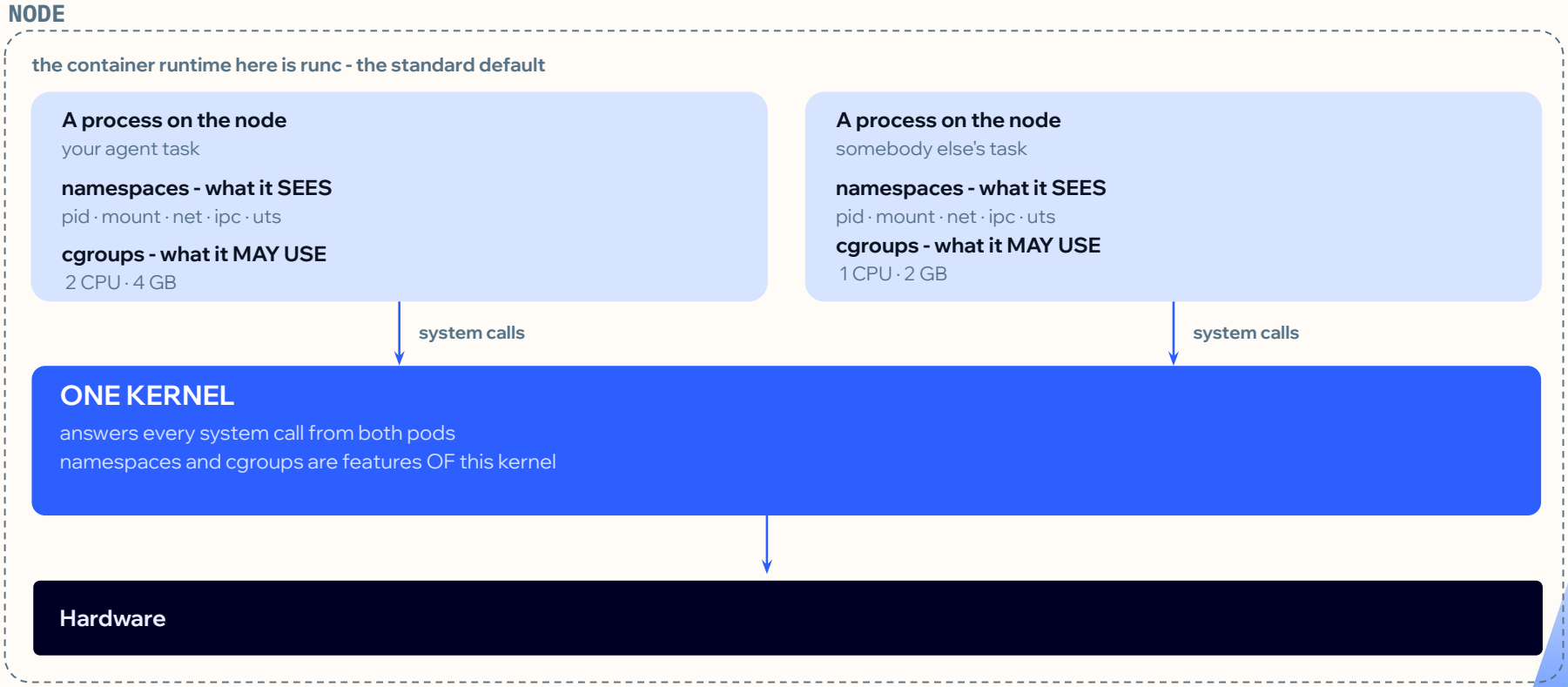


marvik.

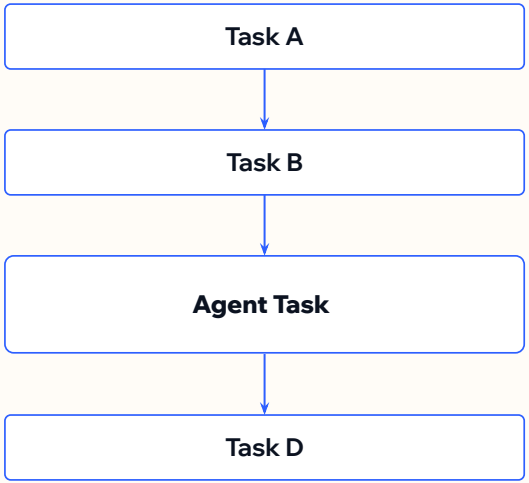
How Airflow runs a task on Kubernetes.



A pod is a container running on a node.



Back to our DAG

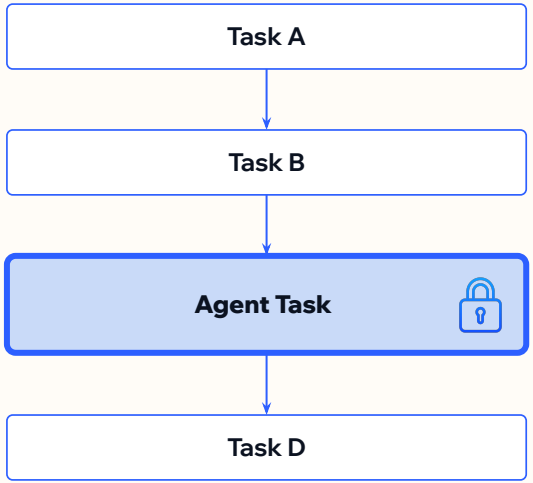


Each task is a pod.

Each pod is a container on a node.

The agent task runs code chosen at runtime, sharing a kernel with every other pod and the node.

Back to our DAG



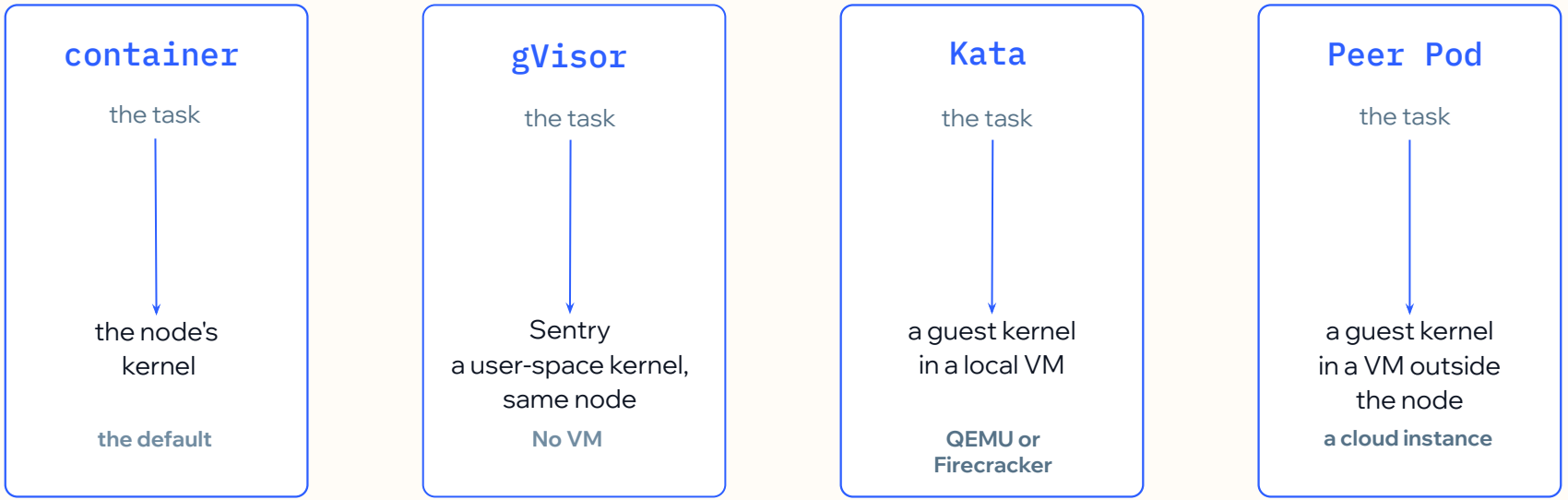
Same picture. One change.

Three of these tasks run code we wrote and reviewed. This one runs code that did not exist when the DAG was written, and it is sharing a kernel with every other pod on that node.

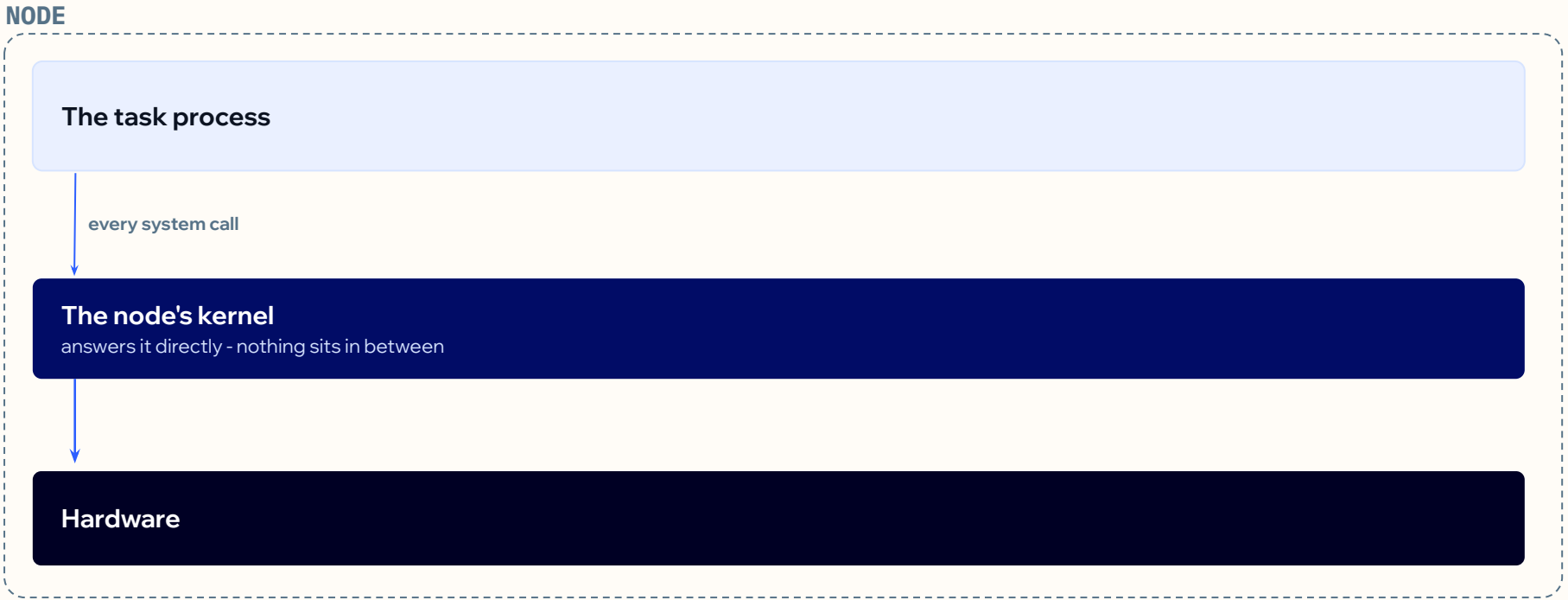
So it needs a boundary. A stronger isolation

Four ways to fill that box

Seccomp and dropped capabilities narrow the request.
They do not change the answer.

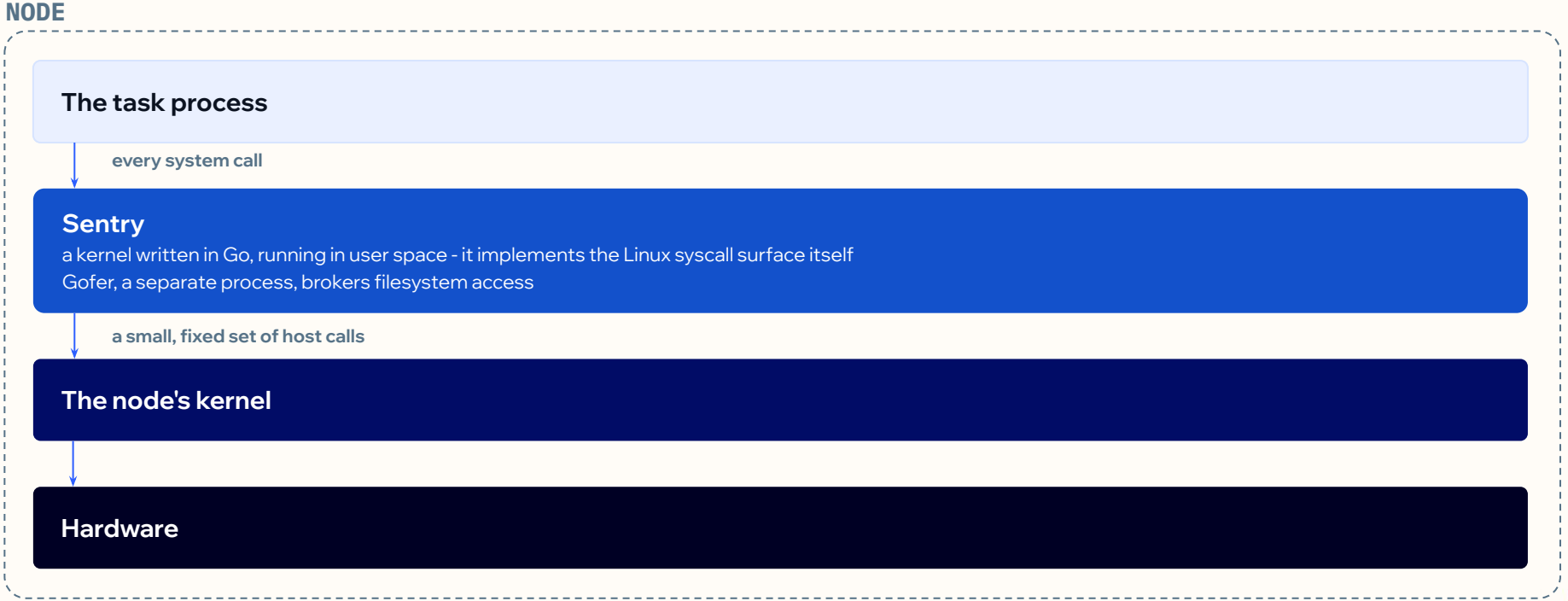


How each one works - Container



A bug in that kernel is a bug for **every pod on the node.**

How each one works - gVisor

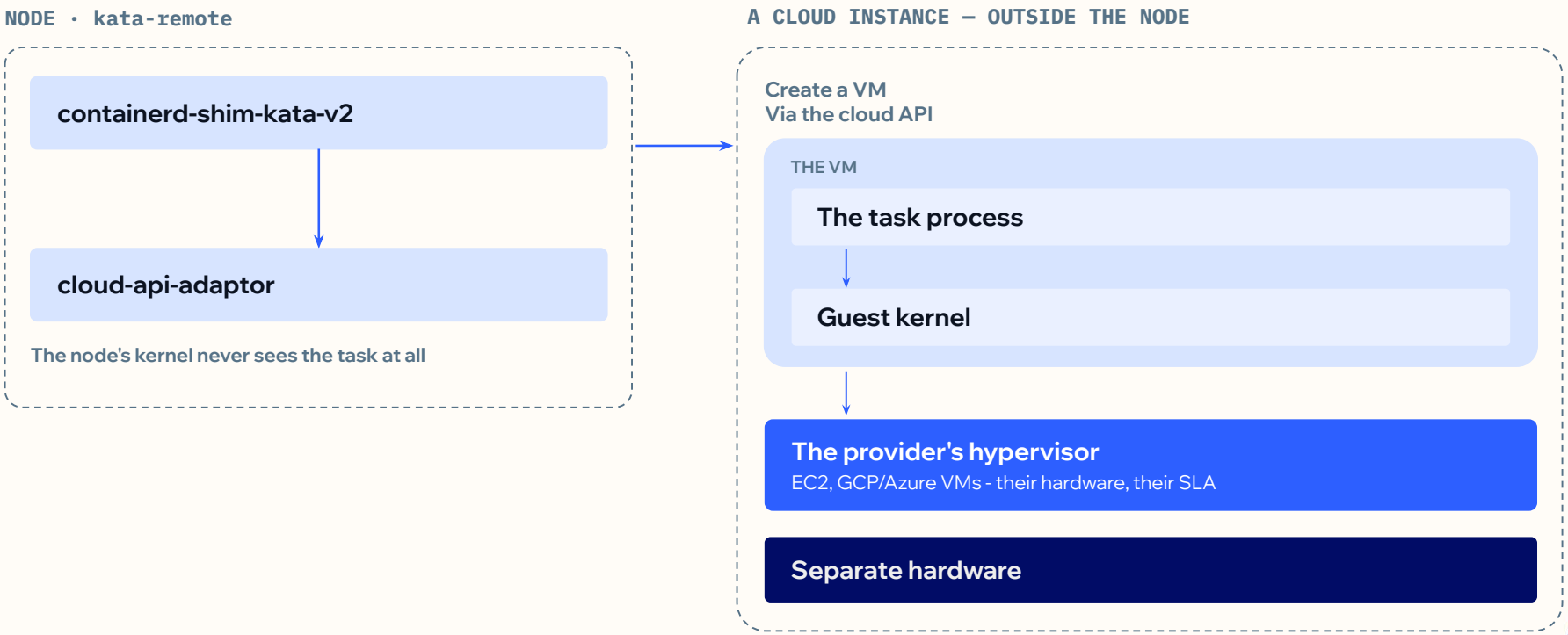


The task never reaches the host kernel directly. No VM, so no KVM and no nested virtualisation.

How each one works - Kata

The guest kernel answers. **To reach the host, an exploit must also cross the hypervisor.**

How each one works - Peer-Pods



No VM inside the node, so no KVM and no nested virtualisation. One extra cloud instance, and its bill.

Switching between them is one field.

Per task. The DAG, the image and the operator do not change.

runc · gvisor · kata-qemu · kata-remote

dag

```python
from kubernetes.client import models as k8s

@task(executor_config={
    "pod_override": k8s.V1Pod(
        spec=k8s.V1PodSpec(runtime_class_name="kata-qemu") # or runsc, or kata-remote
      )
  })
def run_generated_code():
    ...
```

What each option gives up

	WHY YOU'D PICK IT	WHAT IT COSTS YOU
Container	Nothing to install. ~50 ms, a few MB per pod.	No boundary. One kernel bug reaches every pod on the node.
gVisor	Works on any standard cloud node, no KVM, no infrastructure change. Easiest to adopt	Protection depends on Sentry being correct. Incomplete syscall coverage. No GPU.
Kata	Own kernel per pod; escape becomes a two-stage problem. Per-task via RuntimeClass.	Needs KVM - nested virtualisation. ~160 MB per pod.
Peer-Pods	No KVM on the node. Real cloud instance, provider's hypervisor and SLA.	30–90 s to start. A separately billed instance per pod.

The isolation landscape is broader

- **AWS Lambda MicroVMs**: managed Firecracker microVMs designed explicitly for AI-generated/untrusted code; Airflow could call the API via a custom operator.
- **Cloud Run Jobs**: dispatch the task to Google-managed serverless compute; Airflow already has **CloudRunExecuteJobOperator.**
- **AWS Fargate**: run each workload as an isolated managed ECS task; supported by **EcsRunTaskOperator**.
- **Managed batch compute**: AWS Batch / Google Cloud Batch can provision isolated compute per job; both have native Airflow operators.
- **Kata is not tied to QEMU**: current Kata supports **QEMU, Cloud Hypervisor, Firecracker, Dragonball and StratoVirt** as VMM choices.
- **Confidential Kata**: QEMU + **Intel TDX / AMD SEV-SNP** adds hardware-backed memory isolation when the host itself is not trusted.

Airflow does not need to execute the workload itself - an Airflow task can dispatch work into whatever isolation boundary fits the risk.

Real cases

CVE-2025-23266

NVIDIAScape
A 3-line malicious Dockerfile could escape an NVIDIA GPU container and get root on the host. CVSS 9.0

CVE-2022-0185

Linux Kernel
A Linux kernel bug could let code inside a Kubernetes container bypass namespaces and gain control of the node.

CVE-2024-21626

runc “Leaky Vessels”
A malicious image or code already inside a container could exploit a leaked file descriptor to access the host filesystem and achieve complete escape.

Sandbox Escapes

Anthropic / OpenAI
Both have reported agents found ways around isolation controls, exploited unknown vulnerabilities and reached host/cluster-level access

marvik.

Let’s shape the future of AI, together.

We look forward to connecting with you.

Contact us

marvik.ai hello@marvik.ai