KubeCon CloudNativeCon
North America 2026

If it’s shared, it’s vulnerable

Is Kubernetes the right platform for Confidential Computing?



Confidential Containers for GPU Compute: Incorporating LLMs in a Lift-and-Shift Strategy for AI

Zvonko Kaiser - NVIDIA

Containers – chroot on steroids

KubeCon EU 24





KubeCon | CloudNativeCon

North America 2024

From Silicon to Service: Ensuring Confidentiality in Serverless GPU Cloud Functions

Full stack attestation

Zvonko Kaiser NVIDIA

CONFIDENTIAL COMPUTING SUMMIT 2025

Confidential Containers for GPU Compute

Incorporating LLMs in a Lift-and-Shift Strategy for AI

K8s AI/ML platform of choice

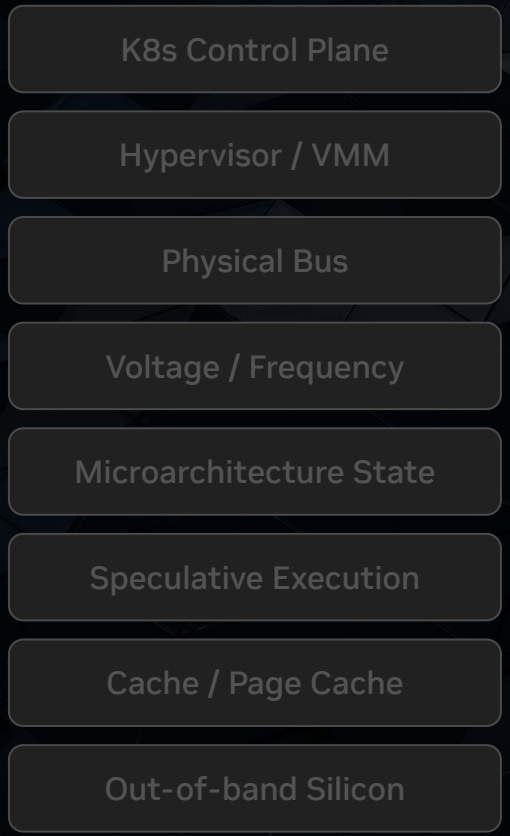
Zvonko Kaiser
Principal Systems Engineer



Trusted AI Anywhere: Kata Containers & Confidential Containers at Scale

Zvonko Kaiser, Principal Systems Engineer | OpenInfra Summit 25

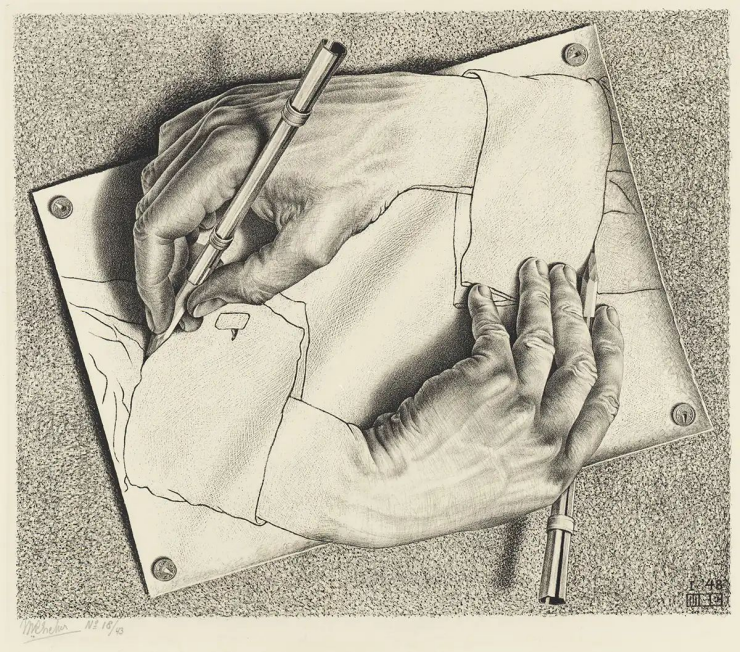
AI/ML on private data is hard



Hardware attestation is necessary. Whether it is sufficient depends on something it cannot measure: **how the infrastructure is shared.**

Two substrates sit beneath every tenant: the silicon and the control plane.

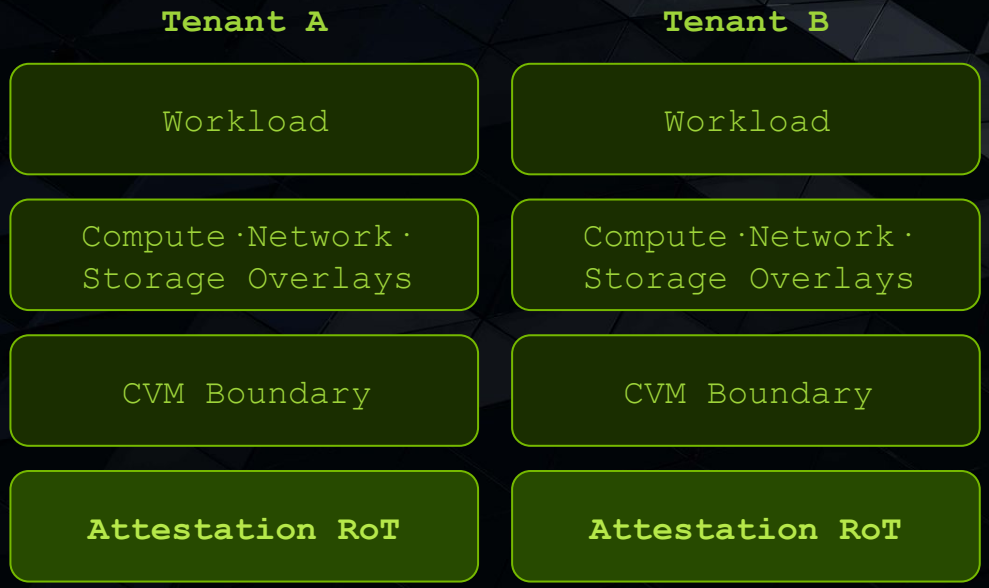
Neither is covered by any tenant's attestation. The operator owns both.



Gödel: any sufficiently complex formal system contains true statements that cannot be proven within the system. Verification requires an external system.

The same applies to runtime integrity. A runtime cannot prove its own integrity using its own facilities.

Confidential computing works the same way. Each component verified by something outside it. The chain terminates in trust.



Each tenant runs independent cryptographic overlays.

Each overlay chain terminates in attestation.

The shared substrates sit beneath every tenant.

The operator owns both.

CONTROL PLANE SUBSTRATE

etcd · Scheduler · API Server

No tenant attestation boundary reaches here

SILICON SUBSTRATE

Silicon · Cache · Bus · Interconnect · Electromagnetic Environment

No tenant attestation boundary reaches here



Each tenant attests their own allocated resources. These surfaces are what no tenant attestation can reach.

Here is what lives in that gap.

Seven shared surfaces. Thirty-nine named attacks since 2023. One layer that is not shared. Zero.

Count: distinct named attacks and assigned CVEs published 2023 to present, per shared resource. Pre-2023 lineage (Spectre, Meltdown, Foreshadow) excluded. Full catalog in appendix.

K8s Control Plane

Hypervisor / VMM

Physical Bus

Voltage / Frequency

Microarchitecture State

Speculative Execution

Cache / Page Cache

Out-of-band Silicon

The Cache is below everything

CACHE / PAGE CACHE · 7 ATTACKS SINCE 2023

Every isolation boundary sits above the cache:
the container, the kernel, the VM , the encryption

The cache is underneath all of them, and it is shared by every tenant on the same socket.

TDXRay (2026)

read an LLM prompt out of it, from one VM to another.

No boundary above the cache can hide what the cache sees.

K8s Control Plane

Hypervisor / VMM

Physical Bus

Voltage / Frequency

Microarchitecture State

Speculative Execution

Cache / Page Cache

Out-of-band Silicon

The encryption was never the hard part

PHYSICAL BUS · 8 attacks · four landed in 2026

TEE.fail
DDR5 bus tap, $1K in parts, pulls keys from TDX, SEV, H100

BreakFAST
Forges SEV-SNP attestation via Infinity Fabric, software-only, every time

Fabricked
Breaks SEV-SNP RMP init via Infinity Fabric, software, every time

NVBleed
NVLink side-channel, reads across VMs on GCP at 88% rate

FROST
NVMe SSD timing from a browser tab, 95% app fingerprinting

ModelSpy
Reads model architecture through a wall via EM, 97.6% rate

PMPlease
RISC-V PMP bypass via DIMM SPD metadata

None of these break the crypto. They share the bus.

K8s Control Plane

Hypervisor / VMM

Physical Bus

Voltage / Frequency

Microarchitecture State

Speculative Execution

Cache / Page Cache

Out-of-band Silicon

The hypervisor cannot be removed

HYPERVISOR / VMM · 9 attacks since 2023

BadAML (2025)

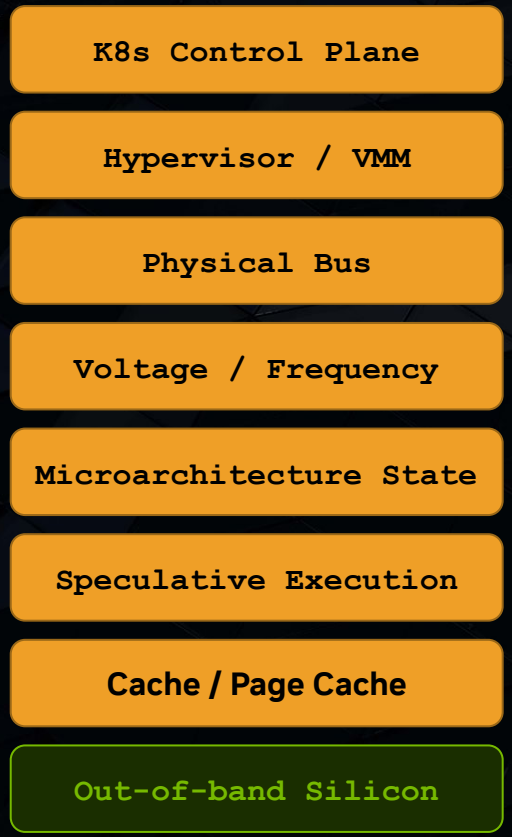
The host injects ACPI bytecode through fw_cfg. It runs inside the guest kernel and patches private pages.

No TEE was bypassed.
Attestation was intact.
The operator needs only the QEMU command line

The CVM has to boot through the hypervisor.

Hence this interface can never be removed.

Heckler · WeSee · SIGY · SEVered · RMPocalypse
Five more in the same class. The boot path is shared



Seven layers. One is different.

CVE-2024-1086 · CVE-2023-0386 · CVE-2023-2640
Container escape, overlayfs, nf_tables · 3 total

BadAML · Heckler · WeSee · SIGY · RMPocalypse · SEVered
Interrupt injection, boot-path bytecode · 9 total

TEE.Fail · BreakFAST · Fabricked · NVBleed · FROST · ModelSpy · PMPlease — Bus tap
Bus tap, interconnect, NVLink, SSD, EM · 8 total

CacheWarp · PMFault
Rowhammer, fault injection · 2 total

LeftoverLocals · PortPrint · Collide+Power
GPU local memory, port contention · 6 total

TDXDown · Downfall / GDS
Transient execution, gather data sampling · 2 total

TDXRay · CopyFail · DirtyFrag · HyperTheft
Prompt reconstruction, page-cache write · 7 total

No published CVE · SEP · Titan M2
Out of band surface not shared

Safety is not deeper in the stack. It is the one row that is not shared. From here the only move is sideways: stop sharing.

The control plane is the other shared substrate

Dedicated silicon stops co-tenant attacks. It does nothing for a shared Kubernetes control plane.

ONE CONTROL PLANE, EVERY TENANT

No row-level isolation

One etcd holds every tenant's secret

One scheduler places every tenant's Pods

One API server enforces every tenant's RBAC

One leaked credential sees every tenant's cluster

AND IT FEEDS THE SILICON ATTACKS

The silicon attacks need co-location

TDXRay needs the attacker on your cache

A shared scheduler puts them there.

The scheduler delivers the attacker to the silicon

Kubernetes was built for cooperative workloads in one trust domain. Etcd is not a multi-tenant database. The scheduler knows the full topology. The control plane assumes every Pod is friendly.

Two shared substrates: the silicon and the control plane. Both are shared and vulnerable.

One node per tenant removes the first.

One control plane per tenant removes the second.

Three isolation architectures

ENFORCED BY SEPARATION

Physically separate silicon

Apple SEP · Google Titan M2
No shared resource to attack.

0 published attacks.

ENFORCED BY HW/SW

Shared silicon + hw/sw isolation

Intel TDX · AMD SEV-SNP · CCA
Shared cache, bus, interconnect.

39 published attacks since `23

ENFORCED BY TRUST

Reduced surface + operator trusts

AWS Nitro · IBM Secure Exec
Operator inside trust boundary.

0 attacks (the other threat model)

Separate-silicon designs have no published attacks because there is no shared resource to attack. The same conclusion appears in the research: Microsoft Research prototyped Core Slicing (OSDI 2023), dedicated physical cores per tenant, after finding isolation within a shared core cannot be reasoned about.

Separation works only when it is complete. One shared resource is enough to undo it.

Where tenants are mutually distrustful, isolation has to be physical.

Confidential Containers secure one surface. A cluster has multiple.

KUBERNETES SURFACE	EXPOSED	COVERED
Compute (the CoCo overlay)	CVM memory contents, attested	secured
Control plane	Pod specs, secrets, scheduling, etcd	not covered
Network	Traffic, DNS, service mesh	not covered
Storage	Access patterns, mount metadata	not covered
Identity	Service tokens, RBAC bindings	not covered
Telemetry	Timing, resource usage, co-location	not covered

Confidential Container secure the compute boundary.

Everything else is a topology question.

The CVM did everything right. The prompt still leaked.

TDXRay (2026): LLM prompt reconstructed word for word from inside an encrypted TDX VM

WHAT TDXRAY DEFEATED

A Confidential TDX VM

- Memory encrypted (TDX)
- Attestation valid
- Hardware confidentiality boundary intact

No exploit, no privilege escalation, no broken crypto

WHAT THE ADVERSARY READ

From the shared L3 cache

A co-tenant on the same socket watched the tokenizer cache-lines accesses.

Encryption scrambles the data at each address. Not which address, not when.

The prompt, word for word.

The cache is below the encryption, below the kernel, below the VM.
The confidential VM is necessary. **It is not sufficient, only separation is.**

There is no middle ground. Either you share the node, or you don’t.

CO-TENANTS ON THE NODE

Shared silicon

Cache, bus interconnect shared

Every silicon attack applies:

- **TDXRay, TEE.fail, NVBleed, the whole wall**

Separating the control plane does not change it. The silicon is still shared.

The silicon surfaces stay open

ONE TENANT PER NODE

Dedicated silicon

No co-tenant on the cache or bus

The silicon attacks have no one to mount them: no co-tenant to watch.

One surface remains: physical access to the machine.

That one is the provider’s responsibility

Per-tenant control planes, dedicated node pools, separate management clusters: all of it is software rearrangement on top of the same silicon. None of it moves the line.

The only line that matters: do two tenants share a physical machine, or not?

One tenant. One node. One control plane.

One node

removes the silicon substrate:
the shared cache, bus, interconnect

One control plane

removes the Kubernetes substrate:
the shared etcd, scheduler, API

One tenant

means nothing is shared at all.
Both substrates, dedicated.

One tenant means you trust only yourself.

Attestation proves what runs inside the boundary.

Only topology decides whether that boundary is shared.

Three questions before you call it secure

1

Who is shared with whom?

Map every tenant to the silicon and the control plane it shares.
The sharing graph is the threat model.

2

What do you actually verify?

For each shared resource, name the verifier.
If there is none, that is residual risk, not a guarantee

3

Is each gap patchable or structural?

Patchable gaps have timelines. Structural gaps need topology.
Treating them the same is how deployments fail silently.

If you cannot name the verifier for a shared resource, it is not secure/confidential, it is hopeful

"If it's smart, it's vulnerable."

— Mikko Hyppönen

"If it's shared, it's vulnerable."

— Zvonko Kaiser

The topology is the security policy.

Thank you. **Questions?**