LINUX FOUNDATION OPEN SOURCE SUMMIT EUROPE · 2026

Off the Laptop, Into Production: An Open Source Stack for AI Agents on Kubernetes

Brian Hammons

Principal Solutions Architect
AWS · @hammbri

Nirmal Mehta

WW Tech Leader Containers
AWS · @nkmehta

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

01

The State of Agents: Where They Show Up & What Makes Them Different

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

Where we are seeing agents show up

01 · Inside the org

Developers and citizen developers (HR, sales, legal) building and running agents at scale across internal operations and software development

INTERNAL

02 · Inside the product

Agents built into the product to enhance features, streamline workflows, and create smarter user experiences

IN-PRODUCT

03 · Running the platform

Enabling operations, SRE, AIOps, security, and platform teams to manage infrastructure, CI/CD, and platforms at scale.

AIOPS / SRE

aws © 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

What's different from a standard application

Stochastic

The same input can return a different output, so continuous evaluation takes over from one-time testing.

→ addressed in the outer loop

Runs code nobody wrote

The model generates the code that runs, so isolation becomes part of the design.

→ addressed in the inner loop

Decides egress at runtime

The agent picks where to connect while it runs, so network policy must work by name, not by IP range.

→ addressed in the inner loop

aws © 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

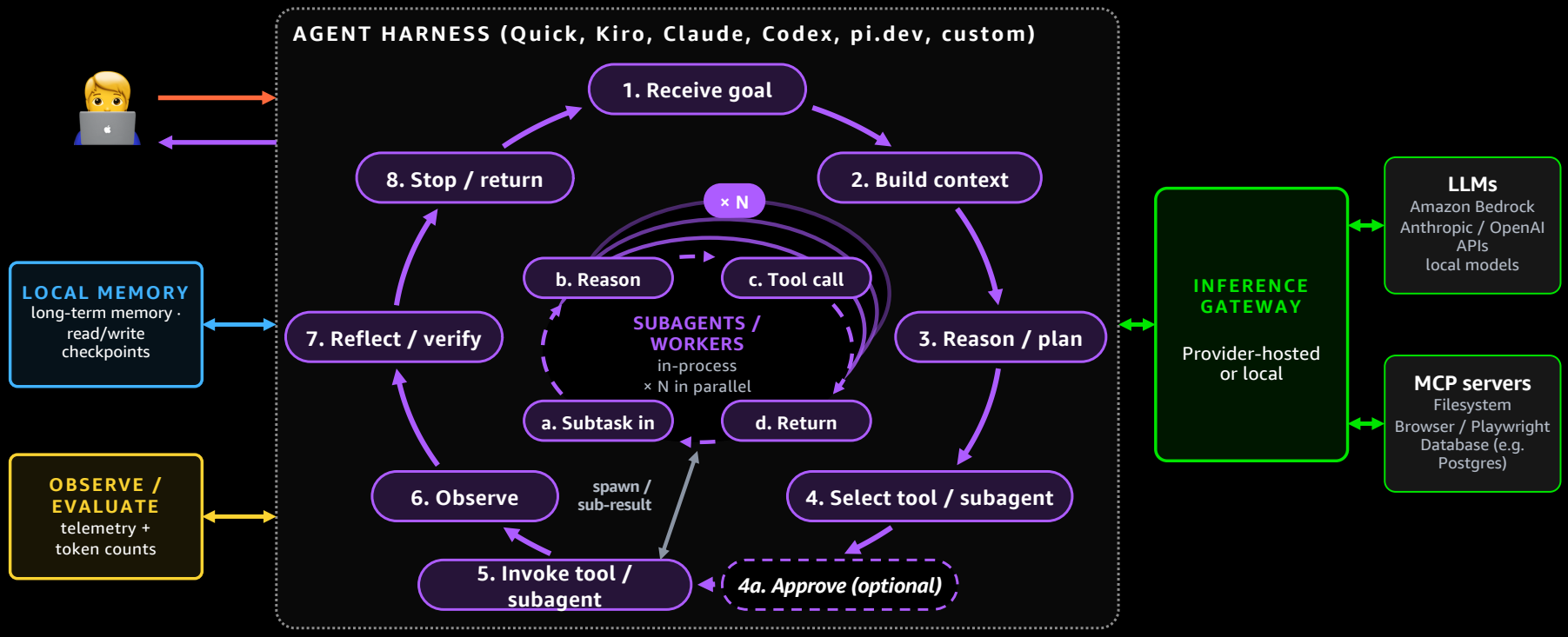
Every team is wiring this differently



...and it often starts on someone's laptop

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

Agent Harness on the Laptop



aws
© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

What we need to run at scale production agents

Both loops deploy onto the one platform you already run

OUTER LOOP

How you operate it

BUILD · DEPLOY · TRACE · EVALUATE · MEMORY

Build	GitLab CE → SCM · CI/CD · registry
Deploy / Dispatch	Argo CD (GitOps) · Karpenter (autoscale)
Observe / Evaluate	Langfuse → tracing, prompts, eval datasets
Memory	Milvus → vector DB / RAG memory

INNER LOOP

Orchestrating Isolated Agents Establishing Guardrails

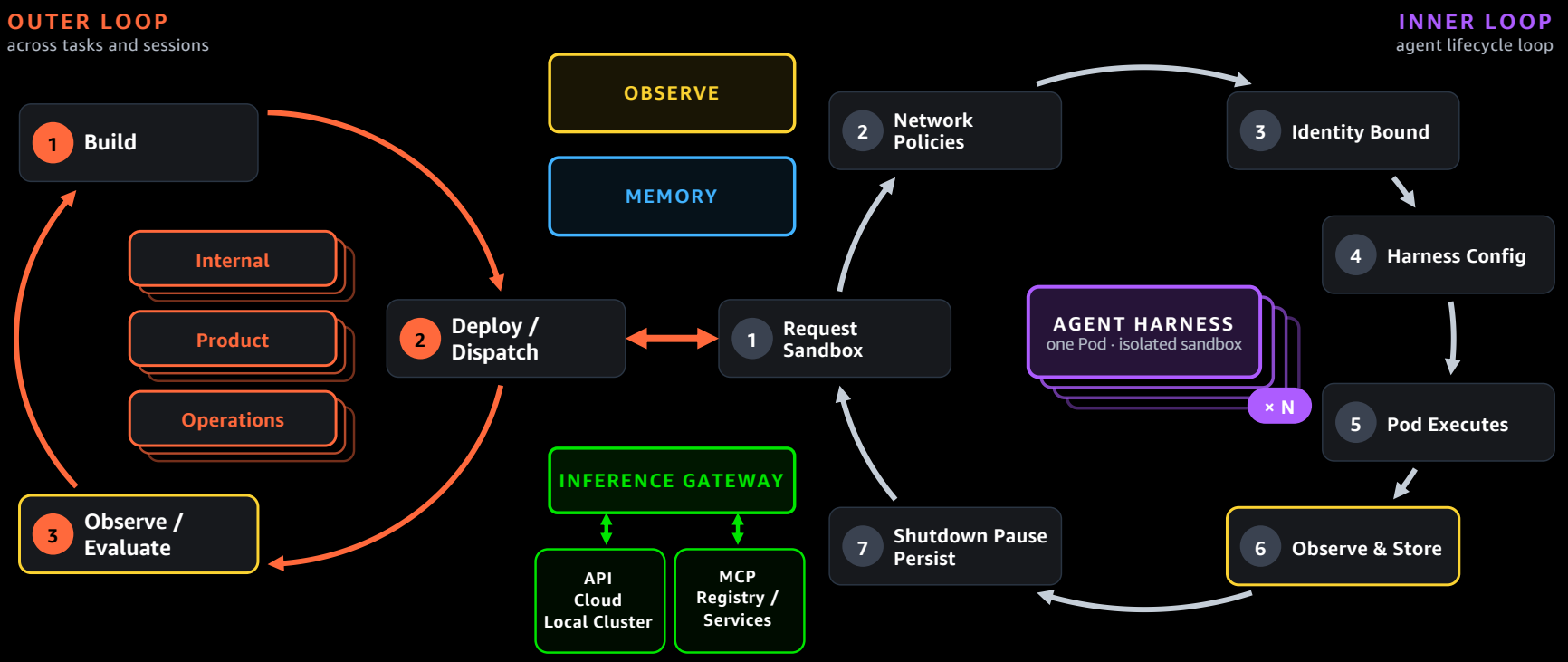
Runtime	runc / gVisor / Kata + Firecracker
Lifecycle	SandboxWarmPool + SandboxClaim
Egress	Cilium FQDN · Native ANP domain names
Identity · Access	SPIFFE / on-behalf-of · MCP gateway

SHARED BASE

Infrastructure: Kubernetes · VPC · Karpenter · Argo CD · ACK

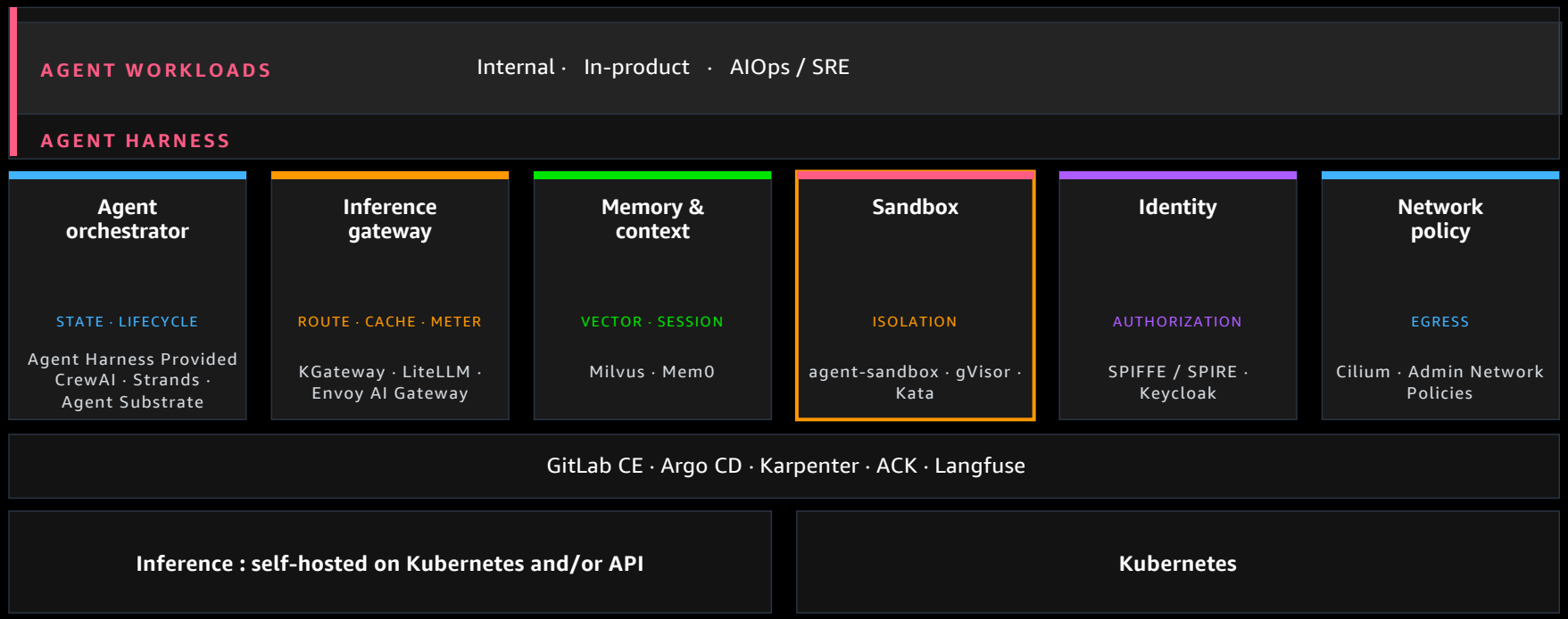
aws © 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

Let's get (Control) loopy! Going beyond our laptops



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

The components needed to scale into production



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

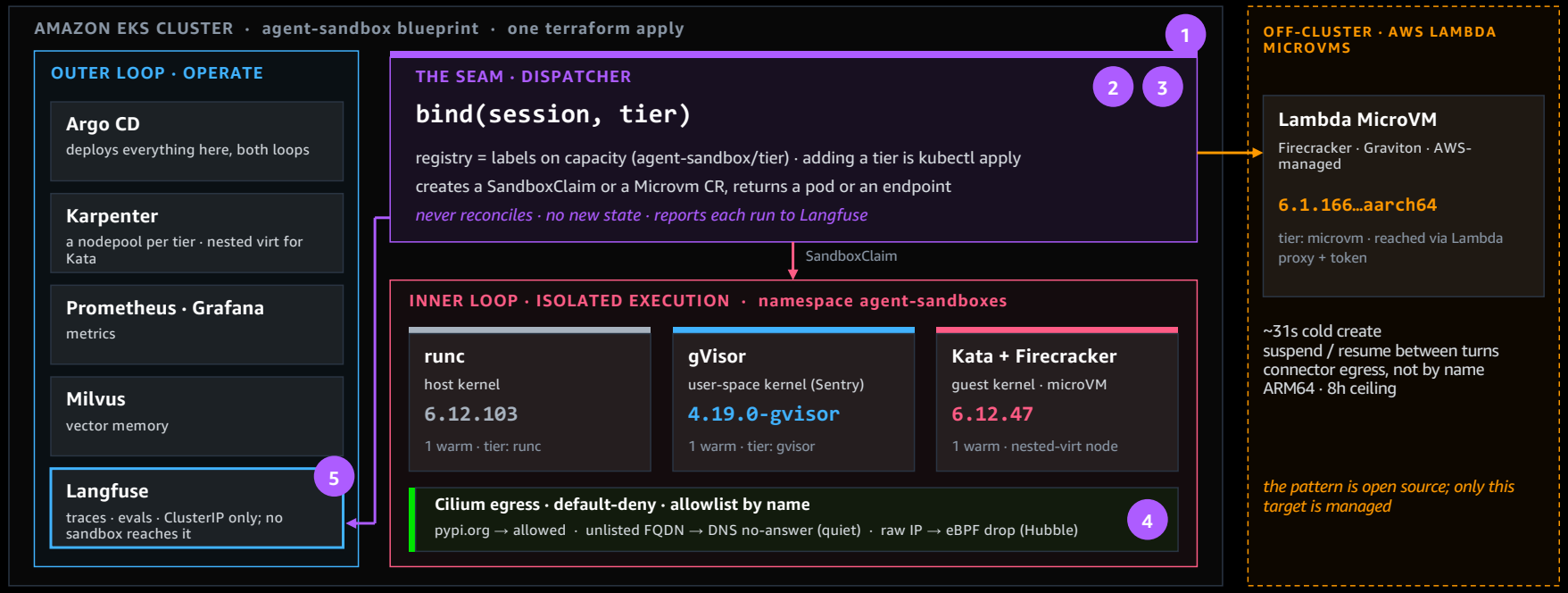
02

Inner Loop: Isolated Execution

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

LIVE · WHAT WE'RE ABOUT TO BUILD

One cluster, both loops, four isolation targets



① one install → ② four kernels, one bind() → ③ warm vs cold → ④ allow / deny at two layers → ⑤ the dispatcher reports the run to Langfuse

Everything shown is deployed and measured on this cluster · awslabs/ai-on-eks · Apache 2.0

Choosing the boundary

User-space isolation and a microVM are different classes, not two points on one line. Choose by what the code may do; warm-pool the latency away.

	← DENSITY · ISOLATION →			OFF-CLUSTER
	runc	**gVisor**	**Kata + Firecracker**	**Lambda MicroVM**
Boundary	shared host kernel	user-space kernel · blast-radius reduction	guest kernel per pod · VM-class	guest kernel per VM · VM-class, AWS-managed
Pick when	code you wrote: orchestration, no execution	model-written code, standard workloads · the default	CPU pinning, custom kernel, nested/privileged · hard multi-tenancy	no nodes to run · bursty sessions · suspend/resume between turns
Warm bind	**<1s**	**<1s**	**<1s**	no pool
Fresh capacity → Ready	**~45s**	**~60s**	**~100s**	**~31s** cold create · sub-second resume
Structurally can't	isolate generated code	pinning · custom kernel · nested/privileged · GPU	EKS Auto Mode · needs nested-virt launch param	FQDN egress · GPU · x86 · >8h · fork-to-N
Egress	FQDN by name	FQDN by name	FQDN by name	connector · all-or-nothing

Bind time was never the question. Choose the boundary first; the pool pays for the latency.

aws © 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

Dispatch beyond the cluster: Lambda MicroVMs

Label the capacity

- agent-sandbox/tier on each pool
- kubectl apply to add new tier

Dispatcher binds

- session → labeled capacity
- pre_bind may veto (admission)

ACK Microvm CR

- controller owns MicroVM lifecycle
- status.endpoint once RUNNING

Return a reference

- no reconciliation loop, no new state
- kubectl get -l session-id=<id>

MEASURED: OFF-CLUSTER BIND

~31s MicroVM CR create → RUNNING + endpoint, round-trip through the Lambda proxy with a port-scoped auth token.

WHY IT MATTERS:

- One interface across all tiers (in-cluster → runc / gVisor / Kata; off-cluster → Lambda MicroVMs)
- Dispatcher never reconciles; agent-sandbox owns in-cluster lifecycle, ACK owns the MicroVM lifecycle
- Relies on the Kubernetes control plane to do the heavy lifting, instead of requiring you to create one from scratch

aws
© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

LIVE · STEP ONE

One install, both loops, one base

① **Platform** — ONE INSTALL
② Tiers — 4 TARGETS
③ Claim — ~1S BIND
④ Egress — ALLOW / DENY
⑤ Trace — THE SEAM

BLUEPRINT.TFVARS · THE WHOLE ENVIRONMENT

```
  # templates · warm pools · claims
enable_agent_sandbox = true
# FQDN egress
enable_cilium               = true
 # the outer loop, watching
enable_kro                  = true
enable_langfuse             = true
  enable_milvus               = true
enable_kube_prometheus_stack = true
# off-cluster tier
enable_ack_lambdamicrovms    = true
```

$ KUBECTL GET APPLICATIONS -N ARGOCD

agent-sandbox	Synced	Healthy	INNER
cilium	Synced	Healthy	INNER
kro	Synced	Healthy	INNER
langfuse	Synced	Healthy	OUTER
milvus	Synced	Healthy	OUTER
kube-prometheus-stack	Synced	Healthy	OUTER

Pre-provisioned Oct 6 (~25 min) · Apache 2.0 in awslabs/ai-on-eks

The outer loop and the inner loop are the same terraform apply.

aws © 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

LIVE · STEP TWO

Four boundaries, one bind()

① Platform	② Tiers	③ Claim	④ Egress	⑤ Trace
ONE INSTALL	4 TARGETS	~1S BIND	ALLOW / DENY	THE SEAM

ONE bind() PER TIER, uname -r executed INSIDE THE SANDBOX

	BIND	KERNEL	
runc	0.82s	6.12.103	the host's kernel: one kernel for everyone
gVisor	0.81s	4.19.0-gvisor	a kernel that isn't there: the Sentry answers syscalls
Kata + Firecracker	0.84s	6.12.47	a real guest kernel: a different machine
Lambda MicroVM	~31s	6.1.166-24.303.amzn2023.aarch64	off-cluster MicroVM, same bind(): CR → RUNNING + endpoint

Same bind() call on four targets. In-cluster isolation went up; the claim didn't get slower.

aws © 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

LIVE · STEP THREE

A sandbox on the next keystroke

① Platform
ONE INSTALL

② Tiers
4 TARGETS

③ Claim
~1S BIND

④ Egress
ALLOW / DENY

⑤ Trace
THE SEAM

WARM · POOL HAS A SANDBOX

```
$ bind session-042 gvisor
```

0.81s

POD

dispatch-worker-gvisor-pool-7d42z

pool-named: adopted

COLD · POOL DRAINED TO 0

```
$ bind session-043 gvisor
```

3.0s

POD

dispatch-session-043

claim-named: created

OPERATE BY LABEL · NO NEW STATE

```
$ kubectl get sandboxclaims -l agent-sandbox/session-id=session-042        # fresh node: ~60s
```

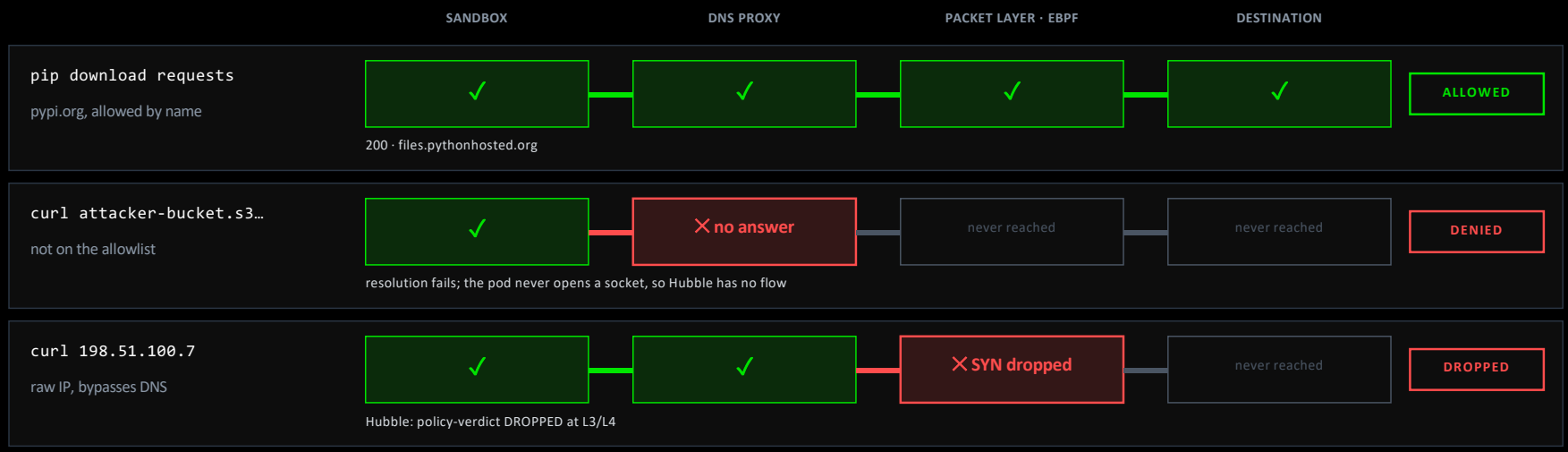
Warm ~0.8s · cold ~3s · fresh node ~60s. The label is the operational surface.

aws © 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

LIVE · STEP FOUR

What can the sandbox reach, and the trap

① **Platform** — ONE INSTALL
② **Tiers** — 4 TARGETS
③ **Claim** — ~1S BIND
④ **Egress** — ALLOW / DENY
⑤ **Trace** — THE SEAM



FQDN answers "where" at DNS. eBPF answers it again at L3/L4. Neither answers "who's asking."

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

LIVE · STEP FIVE

The seam: the outer loop watching the inner loop

① Platform — ONE INSTALL
② Tiers — 4 TARGETS
③ Claim — ~1S BIND
④ Egress — ALLOW / DENY
⑤ Trace — THE SEAM

$ SEAM GVISOR → $ SEAM MICROVM

```
[bind]    tier=gvisor  bind=0.81s
          pod=dispatch-worker-gvisor-pool-95tj2
[report]  langfuse ingestion → accepted (reporter: dispatcher)
          kernel 4.19.0-gvisor
[trace]   FOUND  name=sandbox-run
          session=seam-gvisor-5a9a90  tier=gvisor

$ seam microvm
[trace]   FOUND  tier=microvm
          span output: 6.1.166-24.303.amzn2023.aarch64
```

LANGFUSE · TRACE

sandbox-run

SPAN	tool-call: kernel
OUTPUT	4.19.0-gvisor
METADATA	tier: gvisor · reporter: dispatcher
TAGS	prague-demo · gvisor

Inner → outer: the sandbox did the work; the dispatcher reported it

The sandbox did the work; the platform reported it. Switch tier (even off the cluster) same evidence, different kernel.

aws © 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

03

Best Practices from Production Experience

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

BEST PRACTICES · 1 OF 3

Cold start, and the two barriers that cost us test cycles

SOCI LAZY-LOADING · MEASURED ON EKS

~14s

image pull with SOCI lazy-loading, versus **~3.5 min** cold and unlazy.

SILENT BARRIER #1 · SIZE

SOCI only wires itself up on nodes with ≥4 vCPU / ≥7 GiB. Below that: zero lazy-load mounts, no error, just a slow pull.

Cost: one full test cycle.

SILENT BARRIER #2 · CONFIG

SOCI's default pull mode is not lazy. Without an explicit pull-mode config you get a healthy snapshotter that lazy-loads nothing.

Cost: a second test cycle.

Both failed silently. "Working" and "lazy-loading" looked identical until we measured.

aws © 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

BEST PRACTICES · 2 OF 3

Kata + Firecracker needs nested virtualization

THE GOTCHA THAT FAILS SILENTLY · TWO KATA SURPRISES

WHAT PEOPLE ASSUME

- 7th / 8th-gen instances “just support” microVMs
- Pick a modern family and /dev/kvm is there
- It's a property of the hardware

WHAT'S ACTUALLY TRUE

- Nested virtualization is a launch parameter, not a family property (Karpenter 1.14+ spec.cpuOptions.nestedVirtualization).
- It isn't queryable: the EC2 API has no nested-virt attribute — we checked the whole model — so the family list is hand-maintained and the node fails fast if /dev/kvm is missing.
- A Kata node runs two snapshotters (overlayfs for runc, devmapper for Firecracker), so discard_unpacked_layers must be off or the second unpack fails.

Both bit us once. Both are one line of config — if you know the line.

aws © 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

BEST PRACTICES · 3 OF 3

Where scale bites, and where observability goes blind

THREE YOU'LL HIT · MEASURED OR QUALITATIVELY

Fan-out contention

Distinct-image fan-out on one node is flat to ~10, knees at ~15, ~3× slower by 20.

Burst capacity is a per-node number. Size pools to arrival rate, not average utilization.

DNS vs packet-layer egress

DNS-layer filtering is readable but bypassable; packet-layer is enforcing but quiet.

Two failure signatures. Watch only one and you'll misdiagnose the other — exactly step ④.

Scheduler ceilings

Density is bounded by sandboxes per node, not raw cores — pods, PIDs, snapshotter mounts.

Plan capacity on sandboxes, not cores.

Everything here is publicly verifiable in awslabs/ai-on-eks.

aws © 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

04

The Opportunity Ahead

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

Designing for the agentic ~~future~~ today

Design for Composability

Don't design towards perfection. This is all moving fast, pick composable pieces so you can adapt quickly when the tooling shifts.

→ swap components, don't rewrite

Platform not framework

Agents need a platform, not just a framework. Inner loop and outer loop both land on the same platform you already run. Kubernetes handles isolation, identity, networking, and lifecycle.

→ one platform, both loops

Allow for Flexibility

Agentic architectures give you real speed gains, but we're still figuring out what works. Try things, ship what sticks. Good time to jump in.

→ try and iterate fast

aws © 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

Opportunities to get involved today

PROTOCOLS	MCP AAIF · A2A AAIF · V1.0 · AGENTS.md AAIF
AGENT + INFERENCE GATEWAYS	agentgateway AAIF · Agent Router AAIF · kgateway SANDBOX · LiteLLM · vLLM · Inference Extension SIG NETWORK
ISOLATION · LIFECYCLE	agent-sandbox SIG APPS · V1BETA1 · Kata Containers OPENINFRA · gVisor · Firecracker
MEMORY	Milvus AAIF · Mem0
IDENTITY	SPIFFE · SPIRE GRADUATED · Keycloak INCUBATING · OPA GRADUATED · Cedar SANDBOX · Dogwood
OBSERVABILITY	OpenTelemetry GRADUATED · Langfuse
COMPOSITION · DELIVERY	kro SIG CLOUD PROVIDER · ACK · KubeVela INCUBATING · Argo GRADUATED · Karpenter SIG AUTOSCALING
ORCHESTRATION · AGENT SUBSTRATE	Strands · Agent Substrate SANDBOX · Kubernetes AI Conformance CNCF · Kubernetes GRADUATED

LF · AGENTIC AI FOUNDATION · CNCF · KUBERNETES SIGS

aws

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

Learn more and try it out today!

Read it! Run it! PR it!

github.com/awslabs/ai-on-eks

Agent sandbox blueprints

Join a workshop & try it!

Agents on EKS Workshop
https://s12d.com/er-oss26-eks

Get Hands On with Agents on Amazon EKS

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

Thank you!

Feedback is always welcome

AWS Booth Floor 2 (right outside the keynote entrance)

Wednesday Oct 7
Meet the Experts Nirmal and Brian : 14:15– 14:45
EKS / AI on EKS Agent Sandboxes with Brian : 15:00 – 15:30
Open Agent Platform with Matthew Meckes : 15:45 – 16:15
Lambda: 16:30 – 17:00

Thursday Oct 8
Open Agent Platform with Matthew Meckes : 10:35 – 11:00
Running IDEs Natively on K8s : 15 - 15:30
EKS/AI on EKS Agent Sandboxes with Brian: 16:15 – 16:45

Friday Oct 9
Open Agent Platform with Matthew Meckes : 14:15 – 15:15

Brian Hammons
Principal Solutions Architect · AWS · @hammbri

Nirmal Mehta
WW Tech Leader - Containers · AWS · @nkmehta

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

Q&A

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.