KubeCon CloudNativeCon North America 2026

#KubeCon #CloudNativeCon

Signed, Sealed, Denied: Attestation-Driven Admission with Kyverno + CEL at AT&T

Chad Swenson, AT&T | Sagar Kundral, Nirmata



Who we are



Chad Swenson

Senior Principal Software Engineer

AT&T

Platform engineering architect and product owner.
Production Kubernetes since 2016.
Kubespray / SIG Cluster Lifecycle.



Sagar Kundral

Solutions Architect

Nirmata

10+ years in cloud native and container orchestration, specializing in Kubernetes and its ecosystem.

Three questions for today

1 **Why isn't signing enough?**

Evidence is useful. Deployment still needs a decision.

2 **How does the platform decide?**

Trust roots, verified claims and destination rules.

3 **How do we live with it?**

Performance, exceptions, rollout and retained evidence.

A valid signature is not deployment approval

Verify the evidence

Signature: trusted signer + artifact integrity

Provenance: where and how it was built

Signed claims: evidence the policy can evaluate

Evaluate the requirements

Apply this destination's requirements

Evaluate the requested image digest

Connect policy, evidence and decision

Why not just enforce at promotion?

The destination matters

The same image can face different environment requirements.

Requirements change

Yesterday's approval is not today's policy decision.

Deployment still needs a gate

Evaluate covered requests from GitOps, pipelines or kubectl.

Promotion approves an artifact.
Admission evaluates this request, against policy now.

Same image. Different destination. Different decision.

ILLUSTRATIVE EXAMPLE

DEVELOPMENT

Same image digest

Supplied: signed provenance; no SBOM

Requires: Verified provenance

ADMIT

PRODUCTION

Same image digest

Supplied: signed provenance; no SBOM

Requires: Provenance + signed SBOM

DENY: SBOM missing

The signature isn't broken. The evidence isn't sufficient.

What we needed from the platform

Shared platform controls

App teams shouldn't rebuild the same controls.

Multiple trust roots

Internal builds and approved upstream sources.

Retained decision evidence

Connect the policy, the evidence and the result.

Engineering constraints

Measured latency | High availability | Deliberate failure behavior | Existing pipelines

Three building blocks

1

Sigstore

Signing identity + trust material

Public or private trust setup

2

GitHub Artifact Attestations

SLSA build provenance

SBOM attestations when generated by the pipeline

Custom test & quality attestations

3

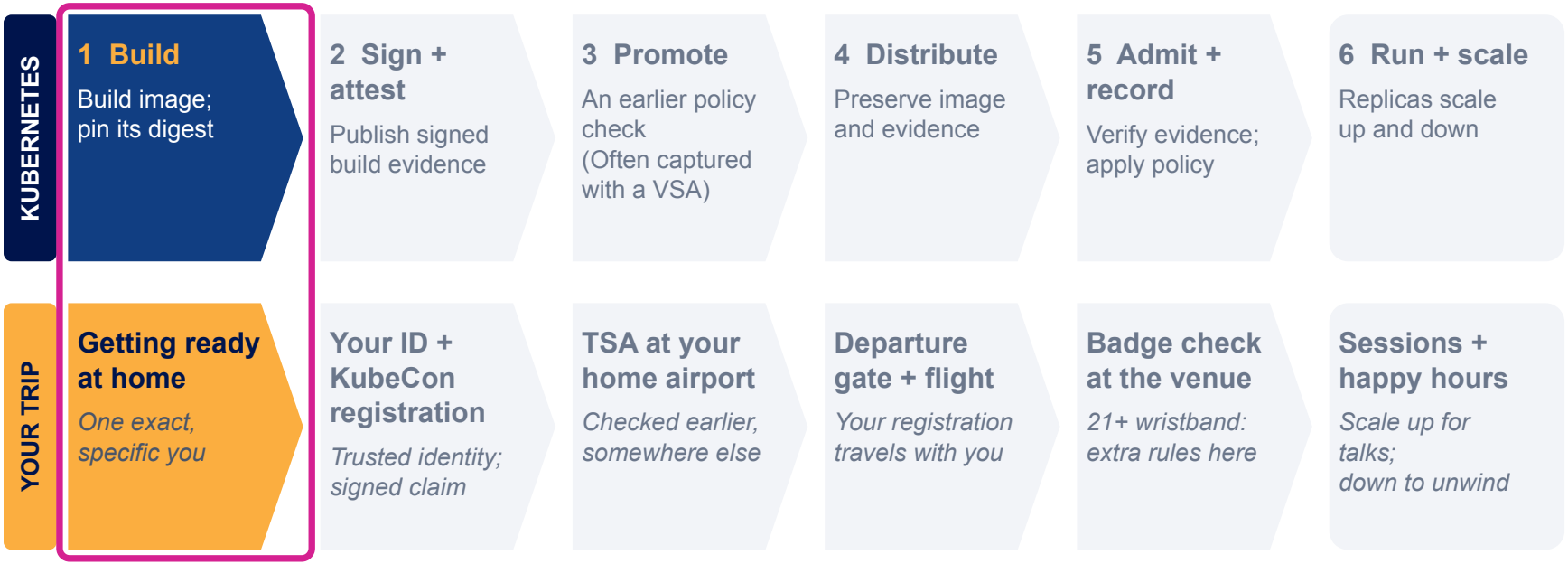
Kyverno + CEL

Verify evidence

Evaluate claims and destination requirements

Provenance = how it was built | SBOM = what's inside | CEL = policy expressions

End-to-end flow: your trip to KubeCon



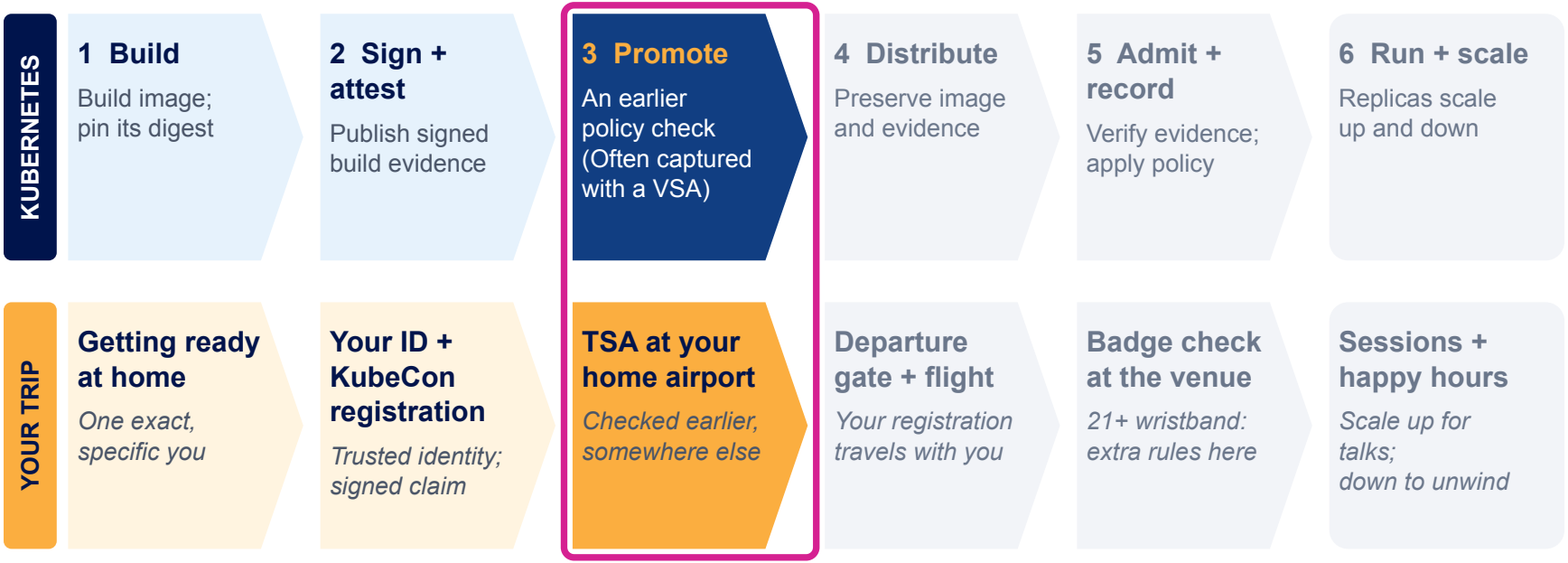
You are the artifact: one exact, digest-pinned you.

End-to-end flow: your trip to KubeCon



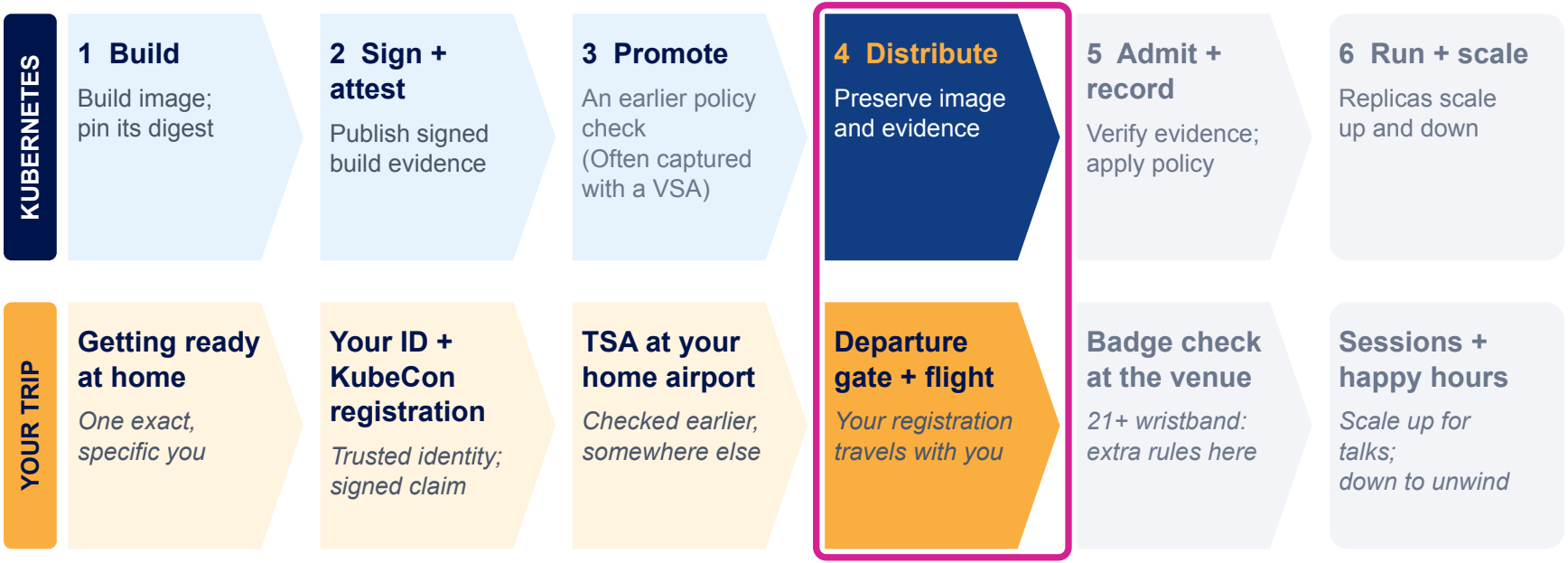
A trusted identity. A signed claim. Not an all-access pass.

End-to-end flow: your trip to KubeCon



Promotion is useful. The destination still has a say.

End-to-end flow: your trip to KubeCon



Your registration only helps if the venue can find it.

End-to-end flow: your trip to KubeCon



Some of you drove. The venue still checks your registration or badge.

End-to-end flow: your trip to KubeCon

KUBERNETES	**1 Build** Build image; pin its digest	**2 Sign + attest** Publish signed build evidence	**3 Promote** An earlier policy check (Often captured with a VSA)	**4 Distribute** Preserve image and evidence	**5 Admit + record** Verify evidence; apply policy	**6 Run + scale** Replicas scale up and down
YOUR TRIP	**Getting ready at home** *One exact, specific you*	**Your ID + KubeCon registration** *Trusted identity; signed claim*	**TSA at your home airport** *Checked earlier, somewhere else*	**Departure gate + flight** *Your registration travels with you*	**Badge check at the venue** *21+ wristband: extra rules here*	**Sessions + happy hours** *Scale up for talks; down to unwind*

One shared control. Applied across deployment requests.

One policy engine. Multiple trust roots.

Internal builds

GitHub Actions signing identity

GitHub private Sigstore trust material

Approved workflow + required claims

Approved upstream images

Public Sigstore or vendor trust

Explicitly approved signer

Evidence requirements by source

Explicit image scope + approved attestors → Kyverno + CEL decision

User feedback became upstream code.

[Jim Bugwadia: private-Sigstore fix #16666 • Kyverno 1.19]

Trust the evidence. Then evaluate the claims.

POLICY WORKBENCH · VERIFY-PROVENANCE · KYVERNO 1.19.2

CEL EXCERPT: VERIFY BEFORE READING PAYLOAD

```
images.containers.map(image,
  verifyAttestationSignatures(
    image, attestations.slsa,
    [attestors.github]
  ) > 0
    ? extractPayload(
        image, attestations.slsa)
    : dyn(null))
```

1 Verify trusted evidence

Central reusable workflow, checked against GitHub's private Sigstore root.

2 Evaluate verified claims

Built by our org: immutable repository_owner_id == "7470644"

3 Apply destination rules

DEV: provenance, SBOM, vuln scan. PROD: also 2+ code-review approvals.

PROD: DENY · "Image requires at least 2 approving code reviews" | DEV: ADMIT

What changed for our use case

KYVERNO 1.19 → 1.19.2: WHAT UNBLOCKED US

Private Sigstore
Kyverno 1.19.0

Jim Bugwadia's #16666 cosign.trustedRoot for GitHub's private CA.

Cover more workloads
Kyverno 1.19.1

Autogen for custom workload CRDs.
Exception-scope security fix (GHSA-5cjf-wwfg-pj4c).

Lighter admission path
Kyverno 1.19.2

Request context built once per admission.

Running ImageValidatingPolicy in Enforce? Use 1.19.2 or later.

Less repeated work on the admission path

KYVERNO 1.20 · IN v1.20.0-rc.1 (NOV 2), NOT YET GA

1

Reuse compiled policies

IVPOL CEL programs compiled once, reused across admissions (#17599).

2

Skip the registry on a cache hit

A cached verdict skips the registry round trip (#17974).

3

Evaluate concurrently

Matching IVPOLs evaluate in parallel, each with its own context (#17496).

That is the mechanism. The next slide is where we earn the numbers.

Performance: before and after

MEASURED · ONE LAPTOP · RATIOS TRAVEL, MILLISECONDS DO NOT

Same setup. Only the Kyverno version changes.

Metric	Kyverno 1.18.2	Kyverno 1.19.1	main (1.20-dev)
Admission latency p50 / p99	990 / 1170 ms	312 / 333 ms	96 / 107 ms
Admissions/sec (concurrency 1)	1.0	3.2	10.4
Registry calls per admission	14	10	3
CPU / memory per replica	44 mC / 607 MiB	67 mC / 463 MiB	34 mC / 65 MiB

Setup: KinD 1.31 + KWOK fake nodes · 1 IVPOL (cosign key, Fail/Deny) · 1 signed image · warm cache
Load: k6 closed loop, concurrency 1, 120 s · +25 ms registry delay · cells with >1% errors or restarts discarded

Operating it: make the hard choices explicit

Exceptions

Narrow policy / resource / image scope

Owner, reason and enforced expiry

Failure behavior

Rejection ≠ verification unavailable

Timeouts and outage behavior by tier

Coverage + availability

Know which requests are covered

Plan HA; observe latency and errors

Make the answer a query, not a project

TARGET DECISION RECORD - ILLUSTRATIVE, NOT A LIVE EVENT

Image	Same digest as the policy example
Destination	Production / <cluster> / <namespace>
Policy	Evidence requirement / <revision>
Result	**DENY - required SBOM missing**
Evidence	Provenance verified; SBOM absent
Correlation	<timestamp> / <request ID>
Exception	None in this example

Capture

Include allowed, denied and unavailable outcomes.

Retain

TBD: store, period, policy and evidence references.

Query

Explain what was evaluated and why.

PolicyReports show current state. Historical admission evidence needs its own retention path.

Roll out in phases, not in a panic

1 Audit

Observe violations, latency and errors.

2 Remediate

Fix shared workflows. Track adoption.

3 Enforce by tier

Start with agreed lower-risk scope.

4 Expand coverage

Grow deliberately. Own the exceptions.

Ready: tested coverage, owned exceptions, measured latency and failure behavior.

What we'd do again

1

Make the right way easy

Build evidence into reusable workflows.

2

Use the engine you know

Make controls a shared platform capability.

3

Bring maintainers a real problem

A concrete use case can become upstream code.

Three things to take home

1 **Verify the evidence**

Trust explicit identities and evaluate signed claims.

2 **Enforce the requirements**

Apply shared controls for the requested destination.

3 **Retain the decision**

Preserve what was checked, under which policy, and why.

KubeCon CloudNativeCon
North America 2026

Thank you! Questions?

Verify the evidence. Enforce the requirements. Retain the decision.

kyverno.io
sigstore.dev
GitHub Artifact Attestations docs
Jim's private-Sigstore PR: kyverno/kyverno#16666



FEEDBACK QR

TBD

Paste Sched code

Kyverno in a nutshell

Kubernetes-native policy as code: CNCF project, no new language to learn

1

What it does

- Validates, mutates, generates and cleans up resources
- Verifies image signatures and attestations
- Runs as an admission controller and as background scans

2

How teams use it

- Policies are Kubernetes resources: GitOps, RBAC, kubectl
- Policy reports show results per resource
- PolicyExceptions scope waivers tightly

3

CEL-based policy types

- ValidatingPolicy and MutatingPolicy
- GeneratingPolicy and DeletingPolicy
- ImageValidatingPolicy (IVPOL) for the supply chain

Anatomy of an ImageValidatingPolicy

Each block answers one question: which images, trusted by whom, proven how, enforced where

`matchConstraints`
Which resources: kinds, namespaces, operations

`matchImageReferences`
Which images: globs or CEL on the image ref

`attestors`
Who is trusted: Cosign keyless, keys, certs; Notary

`attestations`
What else must exist: in-toto (SLSA, SBOM), OCI referrers

`validations`
CEL rules: verifyImageSignatures, verifyAttestationSignatures

`validationConfigurations`
mutateDigest, verifyDigest, required

`credentials`
Registry auth: secrets, cloud providers

`evaluation / autogen`
Admission and/or background; Pod controllers and custom workloads

KYVERNO 1.19

IVPOL hardening in Kyverno 1.19

Exception scope bypass closed

Image- or value-scoped PolicyExceptions can no longer exempt unrelated images on the same resource.

Advisory GHSA-5cjf-wwfg-pj4c

Digest handling

mutateDigest implemented, verifyDigest enforced, validationConfigurations.required fixed.

Safer policy admission

Empty matchConstraints rejected; namespaced policies can't use globalContext.

Webhooks and autogen

Namespaced webhook routing and name collisions, fine-grained autogen, ephemeral containers, background reports.

Cache correctness

Image-prefetch concurrency fixed; attestation payloads restored on cache hits.

Signature fidelity

Cosign v2/v3 annotation verification corrected; failure details kept in policy results.