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

Senior Customer Success Engineer

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

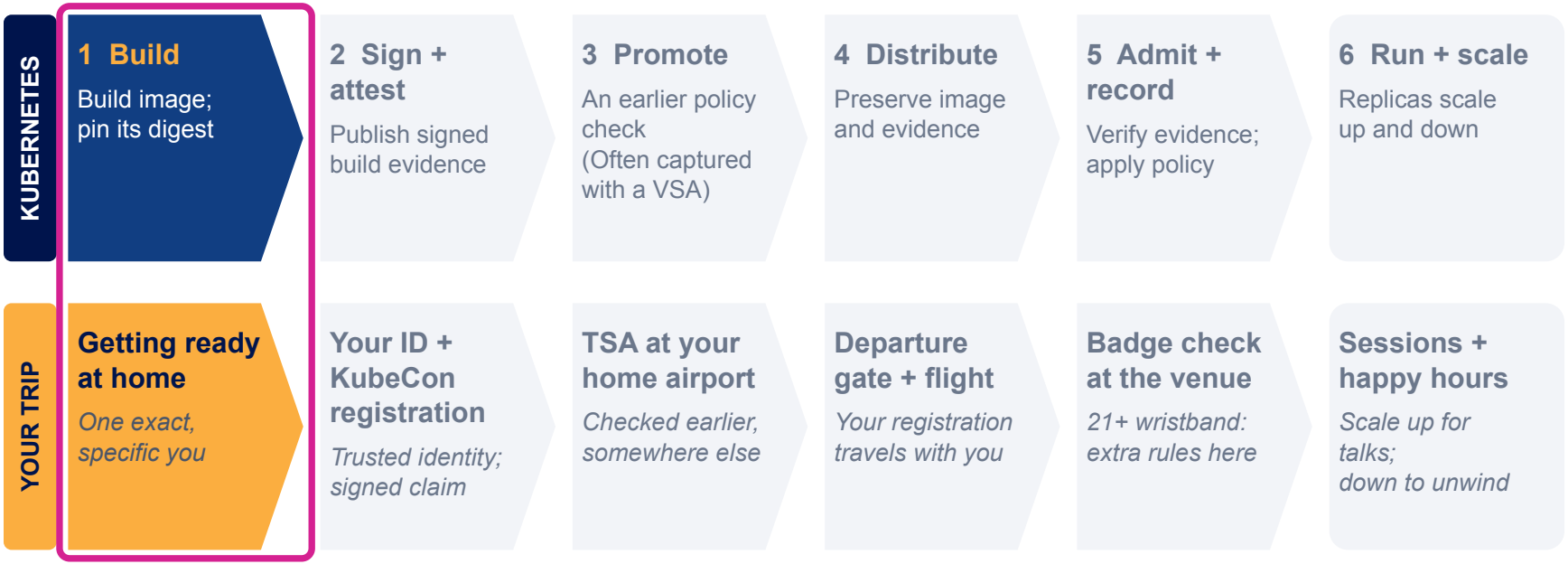
Kyverno + CEL

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



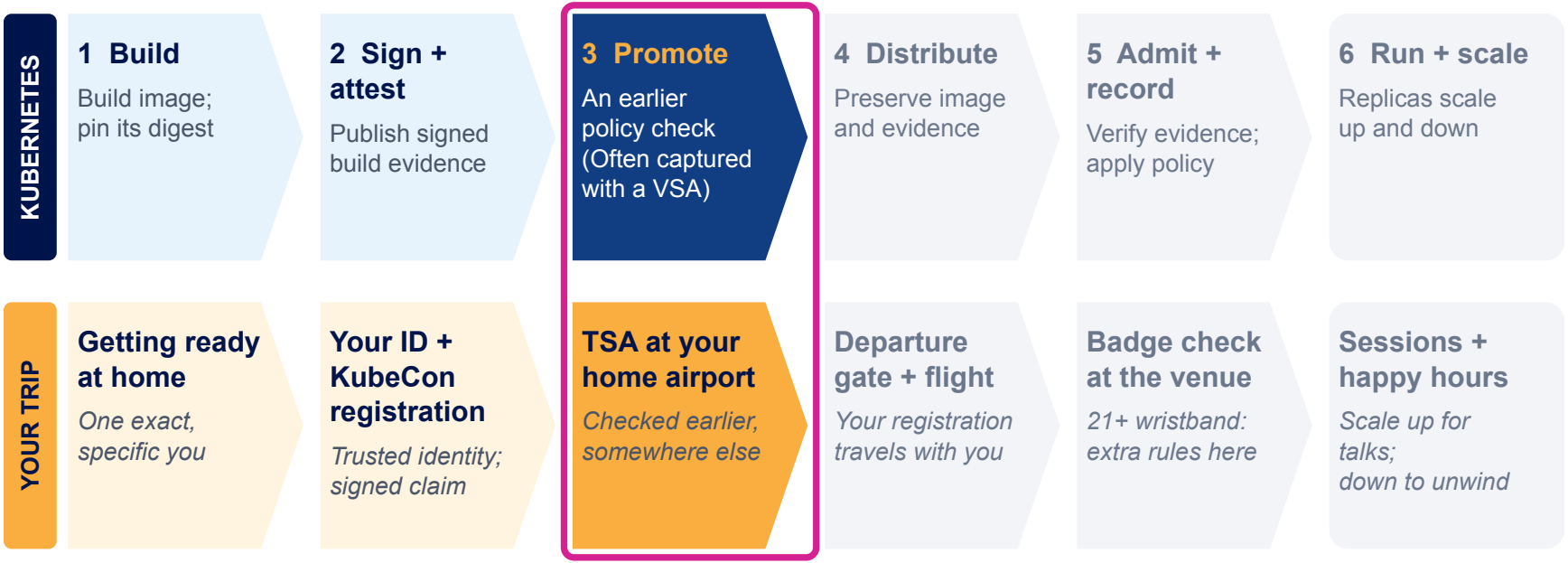
You are the artifact: one exact, digest-pinned you.

End-to-end flow: your trip to KubeCon



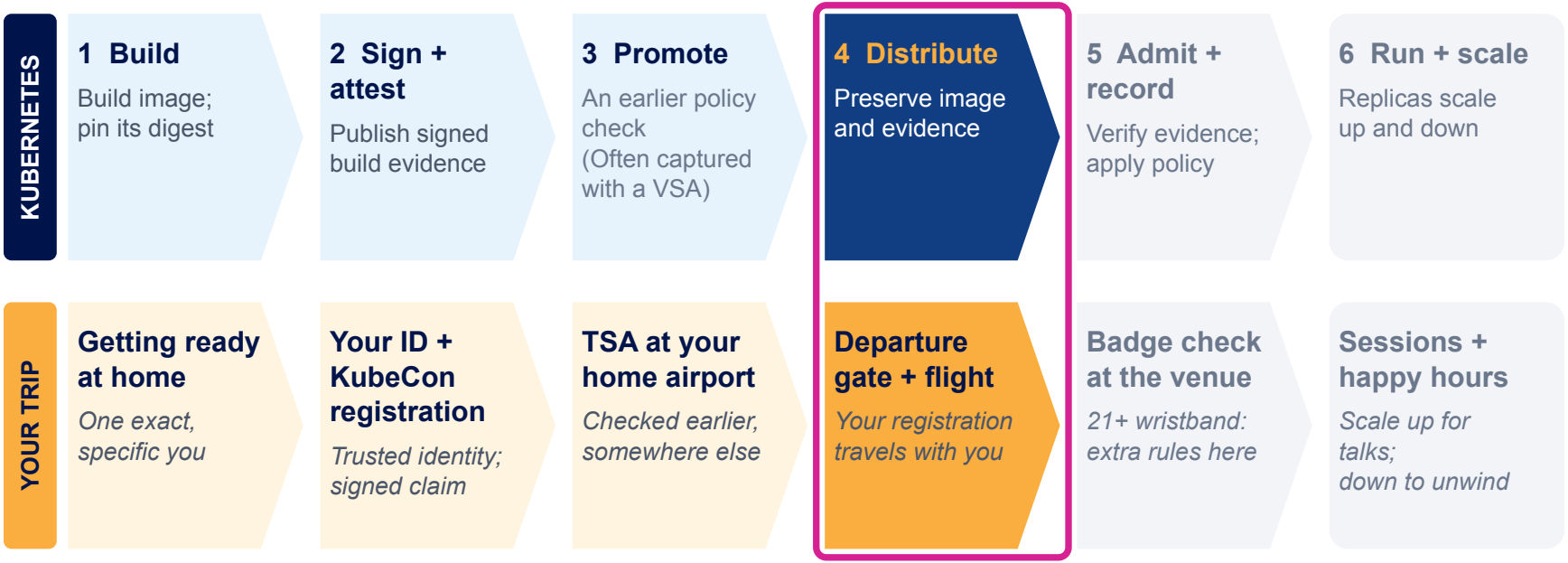
A trusted identity. A signed claim. Not an all-access pass.

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

CEL EXCERPT: VERIFY BEFORE READING PAYLOAD

```
verifyAttestationSignatures(
  image, attestations.provenance,
  [attestors.internal]
) > 0
  ? extractPayload(
      image, attestations.provenance
    )
  : dyn(null)
```

1 Verify trusted evidence

Approved identity + trust material.

2 Evaluate verified claims

TBD: exact claim, value and CEL condition.

3 Apply destination rules

DEV: provenance. PROD: add signed SBOM.

Illustrative production result: DENY - required SBOM missing

What changed for our use case

KYVERNO 1.18 / 1.19 - SAGAR: CONFIRM FINAL RELEASE DETAILS

Private Sigstore / 1.19

Jim Bugwadia's #16666

Custom trust material for private verification.

Workload credentials / 1.18

bb-Ricardo's #15112: ClusterPolicy

IVPOL still needs explicit credentials.

Reuse valid verification results

Caching reduces repeated verification work.

Measure cache behavior with real traffic.

Cover more workloads

Custom-workload autogen

More specific signer-identity matching

Exception-scope fix: 1.19.1. Use a patched, supported release.

Less repeated work on the admission path

1.20 WORK - SAGAR TBD: RELEASE STATUS AT TALK TIME

1

Reuse compiled policies

Across requests; rebuild when policy changes.

2

Share request context

Reuse activation data across applicable policies.

3

Evaluate concurrently

Run applicable policy evaluations concurrently.

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

Performance: before and after

SAGAR TBD - MEASUREMENTS PENDING; NO RESULTS CLAIMED

Same setup. Different version. Show the tradeoff.

Metric	Kyverno 1.18	Kyverno 1.19	Kyverno 1.20
Admission latency p50 / p99	TBD	TBD	TBD
Sustained admission request rate	TBD	TBD	TBD
Verification cache hit rate	TBD	TBD	TBD
CPU / memory per replica	TBD	TBD	TBD

TBD setup: cluster • policies • image mix • cache state • offered load • errors/timeouts

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.