KubeCon | CloudNativeCon India 2026

Why Your Cluster-Wide Policies Are a Risk?

...and what to do about it

Dhruv Puri

KubeCon CloudNativeCon India 2026

Dhruv Puri

Who am I ?

- **DevOps Engineer**, Aspora
- **LFX Mentee** - CNCF Kyverno
- **GSoC Mentee** - Prometheus Operator
- **Community Lead** at Point Blank <. >

Love all things Devops, open-source and cloud ! Keep listening to my same three playlists on loop (ง´ᵕ`)ง♫

BACKGROUND

How we got here: policy-as-code in Kubernetes

Every create or update in Kubernetes passes through admission control.

You can hand-write admission webhooks, but that's a lot of code to build and secure. Policy-as-code fixes that: you declare the rule, and an engine enforces it.

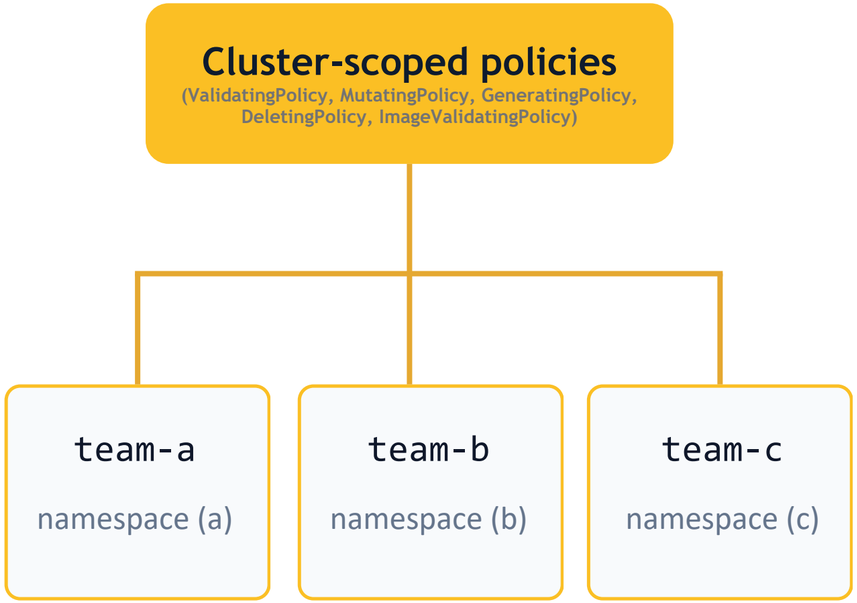
Kyverno is the Kubernetes-native policy engine. Policies are just Kubernetes resources; at admission it can validate, mutate, generate, and verify images.

THE STATUS QUO

What cluster-scoped policies are used for

Initially, Kyverno supported **cluster-scoped** CEL policies: one object that applies to every namespace. That makes them ideal for rules the whole organization must follow:

- **Security baselines** - block privileged pods, enforce non-root, ban :latest images across the cluster.
- **Org-wide guardrails** - require labels, resource limits, and naming conventions for every team.
- **Platform defaults** - mutate or generate shared defaults like sidecars, ConfigMaps, or NetworkPolicies.

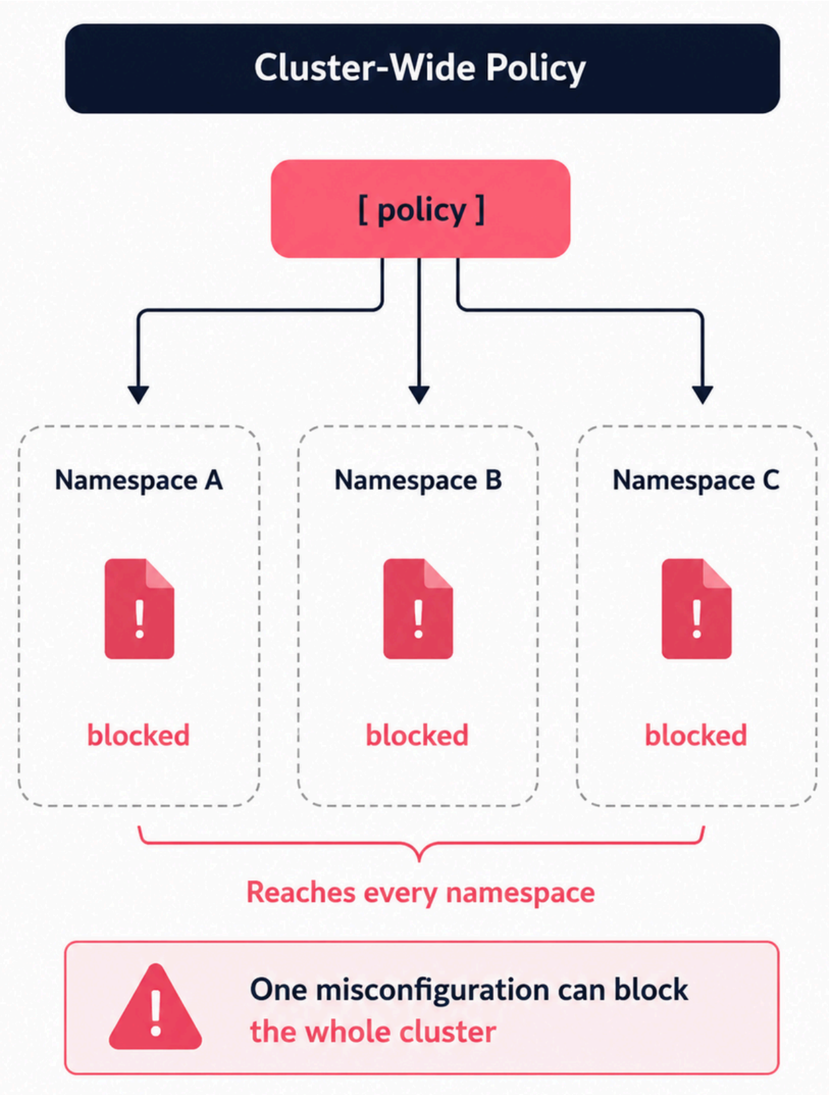


Applies to everyone at once

THE RISK

Four ways cluster-wide scope bites you

1. **Big blast radius** - one misconfigured policy can block or break workloads in every namespace at once.
2. **Heavy permissions** - only cluster admins can manage them, so teams can't self-serve and changes bottleneck.
3. **Slower admission** - every policy is evaluated on every matching request, cluster-wide, adding latency at scale.
4. **Management Overhead** - Namespace owners cannot manage their own policies and any changes or enforcement has to be managed by the cluster admin.



THE FIX

The granular alternative: namespace-scoped policies

The fix is to scope policies to where they actually apply.
My LFX project added namespaced versions of all five CEL policy types with same spec and behaviour as the cluster-wide ones, but each policy lives in, and only acts on, its own namespace. Namespace owners manage their own rules with no cluster-admin rights.

NVPol	NMPol	NIVPol	NGPol	NDPol
Namespaced ValidatingPolicy	Namespaced MutatingPolicy	Namespaced ImageValidatingPolicy	Namespaced GeneratingPolicy	Namespaced DeletingPolicy

Namespace owners self-serve - teams own their rules with no cluster-admin interference !

Reference Links :

1. Kyverno 1.16 release notes
2. Issue #13185 — namespaced policy types

SIDE BY SIDE

Cluster-scoped vs. namespace-scoped

Same CEL syntax on both sides - the difference is who owns the policy and how far a mistake travels.

Cluster-scoped

- Scope: the entire cluster
- Managed by: cluster admins only
- Blast radius: every namespace
- Best for: org-wide baselines

Namespace-scoped

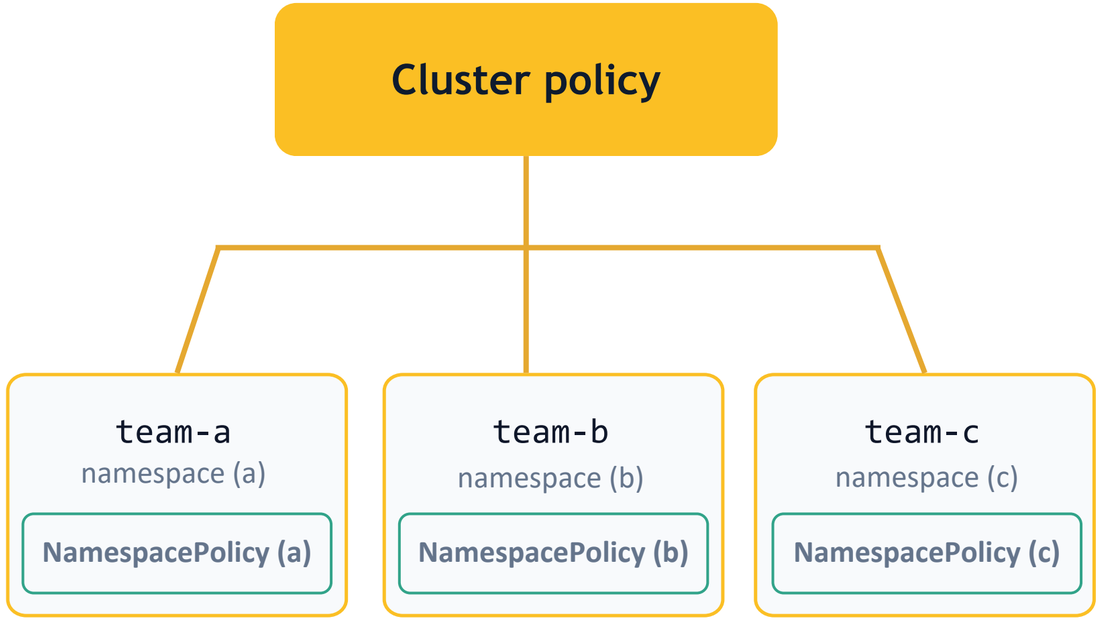
- Scope: certain namespace only
- Managed by: the owner of the namespace
- Blast radius: that namespace team
- Best for: team-specific rules

Policy ownership moves closer to the team and teams control the rules that affect their workloads :)

WHERE IT FITS

When to reach for namespace-scoped policies

- **Multi-tenant clusters** - each team enforces its own rules; a mistake never leaks into another tenant.
- **Developer self-service** - ship guardrails in your own namespace without filing a platform ticket.
- **Progressive rollout** - adopt policy-as-code namespace by namespace, instead of a risky big-bang.



Granular application of policies

TRY IT

What it looks like

A namespace-scoped policy is the same CEL you already know - just with a **namespace** field.

Match the resources you care about, write the rule, and apply.

This governs only Deployments in team-a, nothing else in the cluster.

require-env-label.yaml

```yaml
apiVersion: policies.kyverno.io/v1
kind: NamespacedValidatingPolicy
metadata:
  name: require-env-label
  namespace: team-a
spec:
  validationActions: [Deny]
  matchConstraints:
    resourceRules:
    - apiGroups: [apps], resources: [deployments]
      operations: [CREATE, UPDATE]
  validations:
  - expression: >-
      object.metadata.labels['env'] == 'prod'
    message: "must set env=prod"
```

KubeCon | CloudNativeCon India 2026

ACKNOWLEDGEMENTS

Special thanks to my mentors who guided me throughout the implementation of namespace-scoped policies in Kyverno:

1. Charles-Edouard Brétéché [LinkedIn]
2. Frank Jogeleit [LinkedIn]

IMPORTANT LINKS:

1. https://playground.kyverno.io/#/
2. https://kyverno.io/docs/policy-types/overview/

CNCF Slack :

1. #kyverno
2. #kyverno-dev

LinkedIn ^-^

Thank you !