# Closing the AuthZ Gap in MCP

## Policy-Driven Tool Invocation Control

Oshi Gupta & Sonali Srivastava
Improving

MCP Dev Summit · Mumbai 2026

# Meet the Speakers

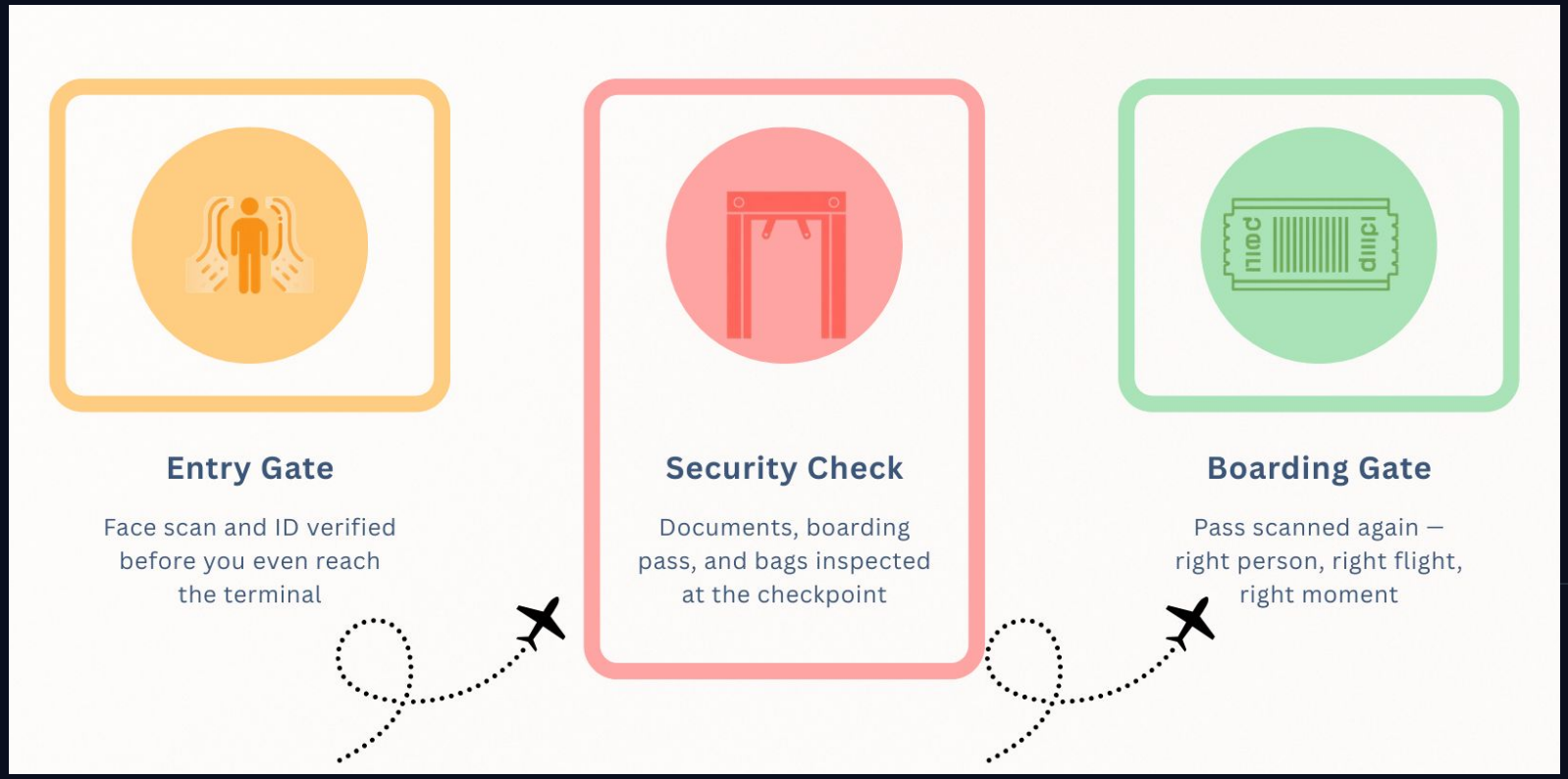
## Oshi Gupta

## Sonali Srivastava

# What We'll Cover

- The MCP auth model and what it leaves unprotected
- Four real threat patterns in multi-tenant agent platforms
- Why Kubernetes RBAC can't solve this
- Policy-as-code with Kyverno: three patterns that close the gap
- Live demo: allowlisting, tenant isolation, human identity injection
- Best practices for rolling this out safely

# Checked at Every Gate




**Right person, right action, right moment?**

# Four Gaps No One Is Watching

## Cross-Tenant Tool Calls

Agent A calls a tool that operates on Tenant B's data. Nothing stops it in current protocol designs.

## Phantom Agent Identity

Audit log shows a service account. The human who typed the prompt is invisible to the backend.

## Unconstrained Parameters

Tool accepts query params for a prod database. Nothing validates them before execution.

## Privilege Escalation Chains

Each agent hop loses context. Nobody can trace whose authority the tool is actually acting under.

# RBAC vs MCP Semantic Context

## RBAC Can Answer

Can this SA list pods in this namespace?

Can this SA read secrets?

Can this SA create deployments?

## RBAC Cannot Answer

Can sre-agent call the delete_record tool?

Does this tenantId match the namespace?

Was this write tool triggered by a human?

**MCP tool calls carry semantic context. RBAC has no vocabulary for it.**

# Policy As A Code

01

## Tool Allowlisting

Denied by default. Each agent SA has an explicit allowlist. Blocked at admission before the MCP server ever sees it.

02

## Tenant Isolation

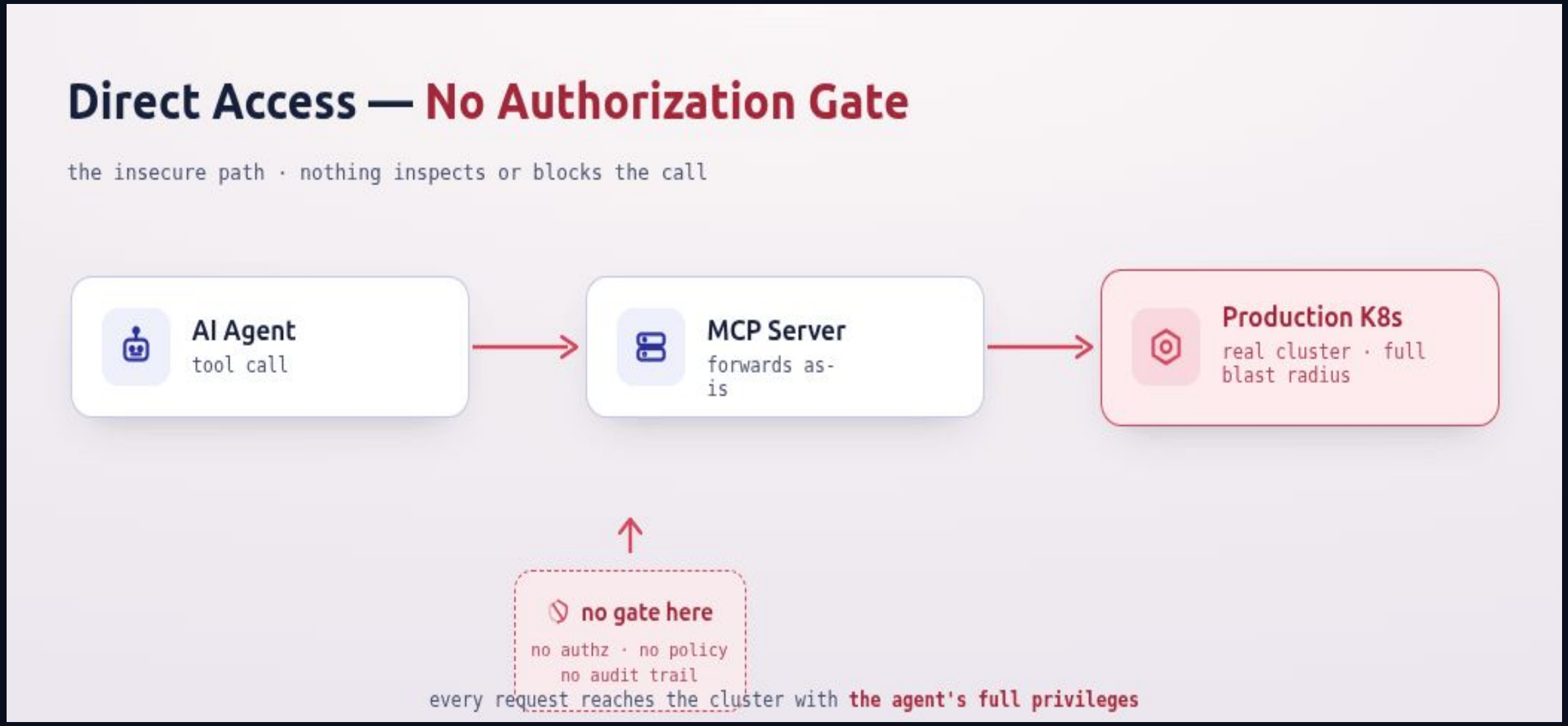
Validates tenantId in the invocation matches the agent's namespace. Cross-tenant calls blocked structurally.

03

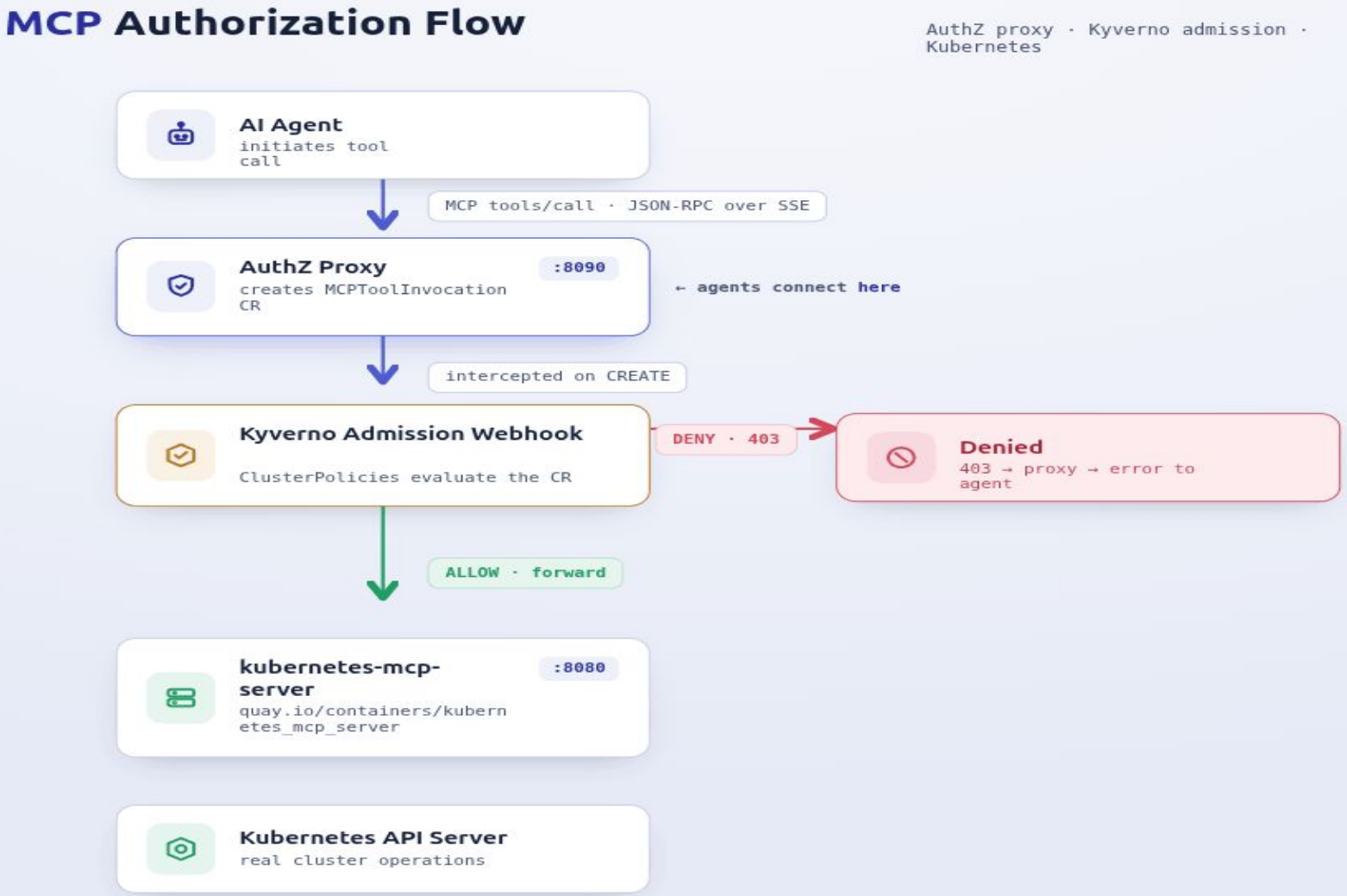
## Human Identity Injection

Mutating webhook writes unalterable triggered-by annotations. Written by Kyverno, not the agent.

# Without Authorization



# How the Stack Is Wired With Authorization




GitHub Repo

# Three Scenarios, Live

DEMO

# Rolling Out Safely

1. **Audit first, enforce second**
   Run Kyverno in Audit mode for a week before writing a single blocking policy.

2. **Spec over Labels**
   Put toolName and tenantId in the spec, not labels, for cleaner policy expressions.

3. **Mutate before Validate**
   Get your audit trail verified before you start actively blocking requests.

4. **Incremental Enforce**
   Flip one policy at a time with 2 weeks of audit coverage per policy.

5. **Human Context Propagation**
   Orchestration layers (LangGraph) must stamp triggering identity. Cannot be retrofitted.

# The AuthZ Gap Is Real

Policy-as-code gives you a concrete, auditable answer to: who authorized that tool call?

Read the Blog

Questions? Connect with us on LinkedIn

**Oshi Gupta · Sonali Srivastava · Improving**