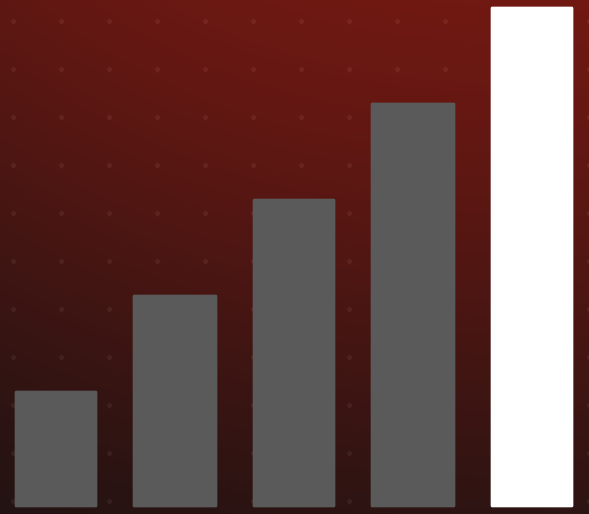
KubeCon | CloudNativeCon
North America 2026

AI INFERENCE + AGENTIC

From Proxy to Platform

Building a multi-tenant AI gateway



PRESS START ▶

Dibyo Mukherjee Lead Software Engineer, Adobe

Tushar More Sr Platform Engineer, Adobe

Adobe

We run the core platform. Now also the front door to models.

OUR DAY JOB

Core platform infra

Kubernetes · Gateways · GitOps

WHAT WE ADDED

One shared AI gateway

One front door to models, for every team

WHO CALLS IT

- Product features
- Internal tools
- Coding agents
- Customer-facing agents

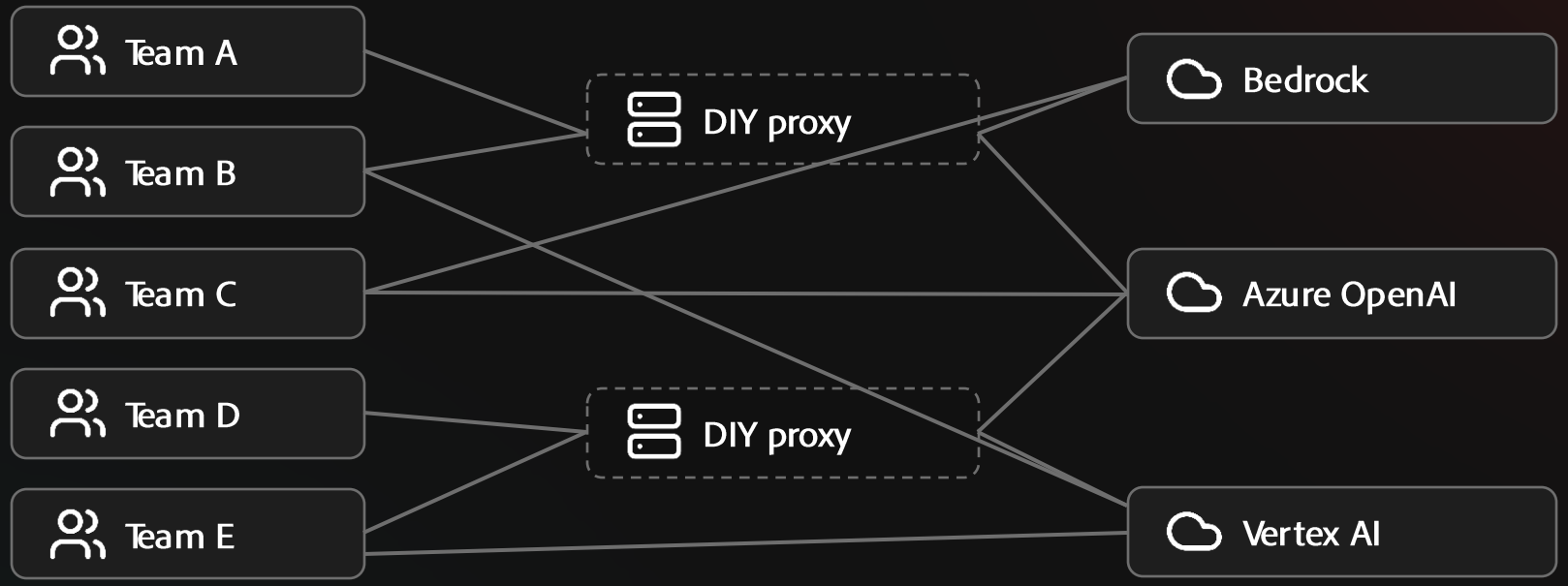
One endpoint

WHAT IT REACHES

- Bedrock
- Azure OpenAI
- Vertex AI

Built on LiteLLM. But everything here applies whatever proxy that sits in the middle.

Pregame: every team ran its own gateway



Who used this key?

What did we spend?

Is it secure?

Own cloud accounts, own keys, a few homegrown proxies. Nobody could answer the three questions.

WORLD MAP

Three things changed the design

PREGAME

Shared tokens

everyone on their own keys

LEVEL 1

Who owns what

when a second team arrives

LEVEL 2

Who's spending

when your users are agents

LEVEL 3

Guarantees

when you're on the critical path

NEXT

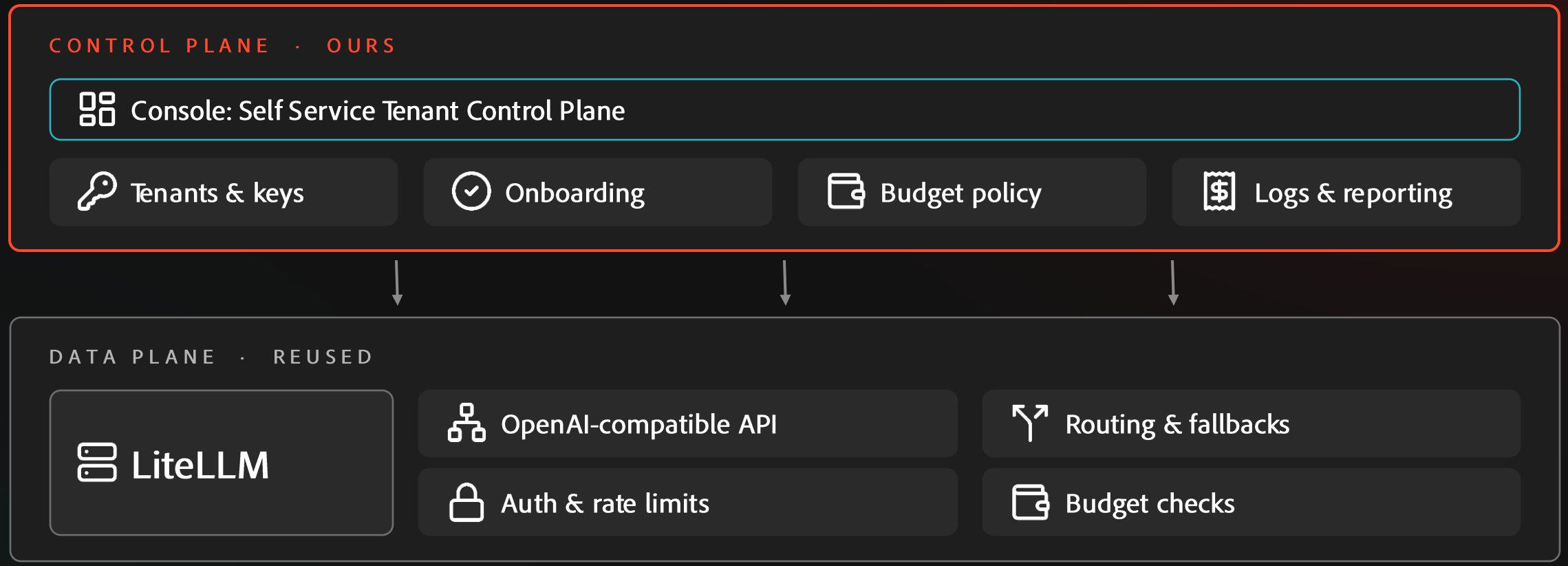
What's next

regions · cost · identity

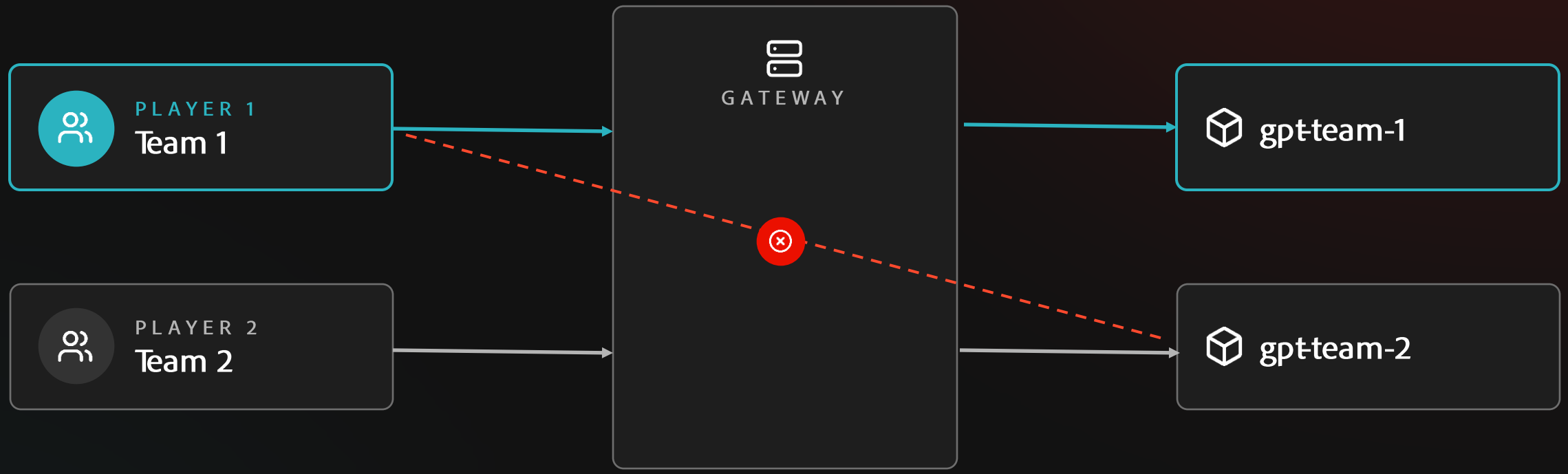
Our path, not a universal maturity ladder.

CHOOSE YOUR LOADOUT

Borrow the engine. Build the controls.



One tenant is easy. Multiplayer changed the rules.



Same gateway, separate lanes. Team 1's key never reaches Team 2's models

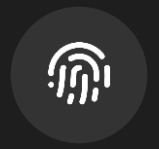
LEVEL 1 · WHEN A SECOND TEAM ARRIVES

The tenant is the team

TEAM = TENANT



Use cases
belong to teams



Service IDs
belong to teams



Budgets
belong to teams

People move between teams. The use case stays. Credentials had to be team-scoped before the first migration could happen.

Teams self-serve. The platform guarantees.

BEFORE a ticket for every change → NOW self-serve in the Console

TENANT-CONTROLLED

- Keys
- Models
- Members
- Budgets
- Fallback rules

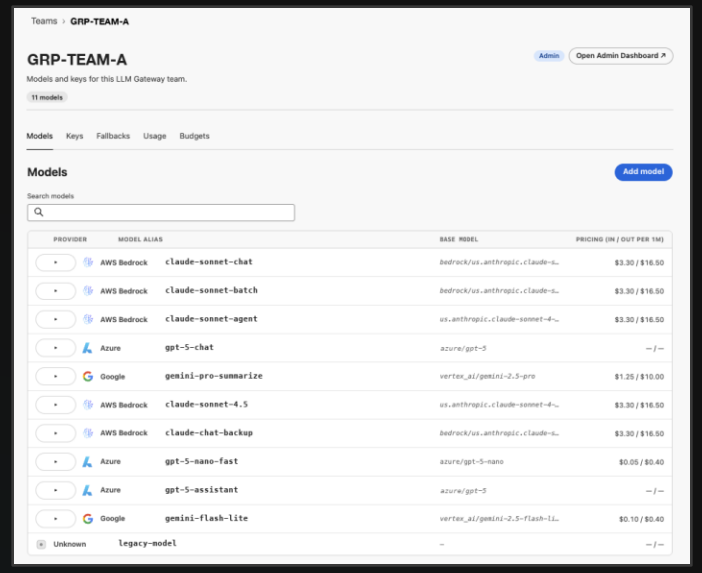
PLATFORM-OWNED

- Logs
- Upgrades
- Regions
- On-call

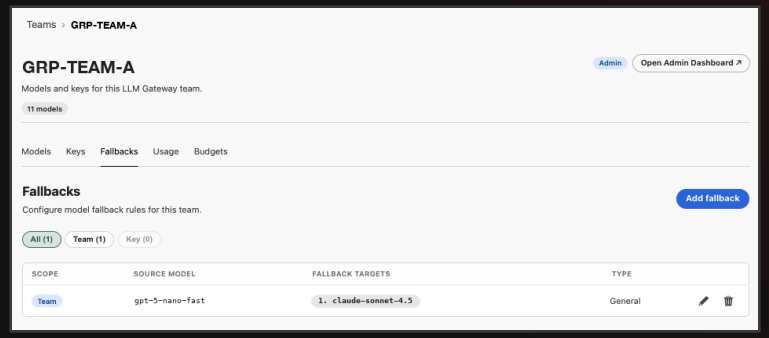
Upstream, and not ours: model approvals, cloud accounts.

Models, fallbacks, budgets. Self-served.

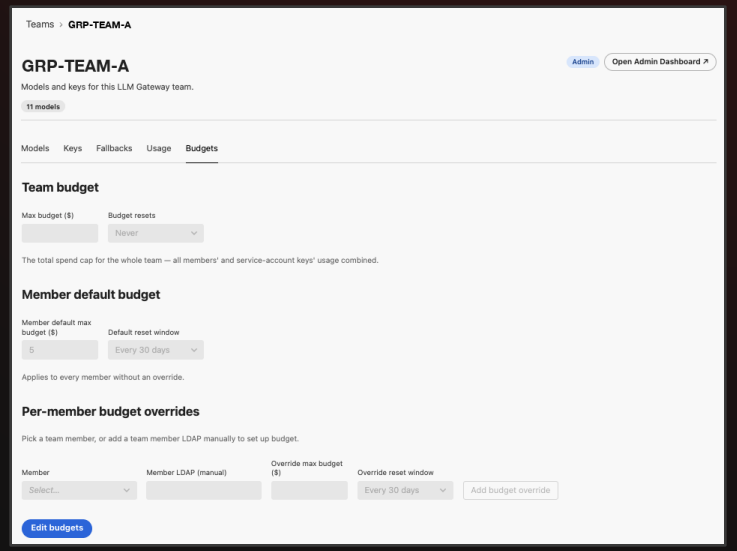
MODELS



FALLBACKS



BUDGETS



Agents changed who's calling.

INTERNAL AGENTS

- Act for one employee
- Mint a new key every turn
- Run with nobody watching

BROKE: The key is the owner

CUSTOMER-FACING AGENTS

- Act for many end customers
- Share one team key
- One heavy customer can drain the team

BROKE: The team is the spender

Internal agents: the key isn't the owner.

WHAT HAPPENED

A new key every turn; each started at $0



Agent generated short-lived keys had no owner.

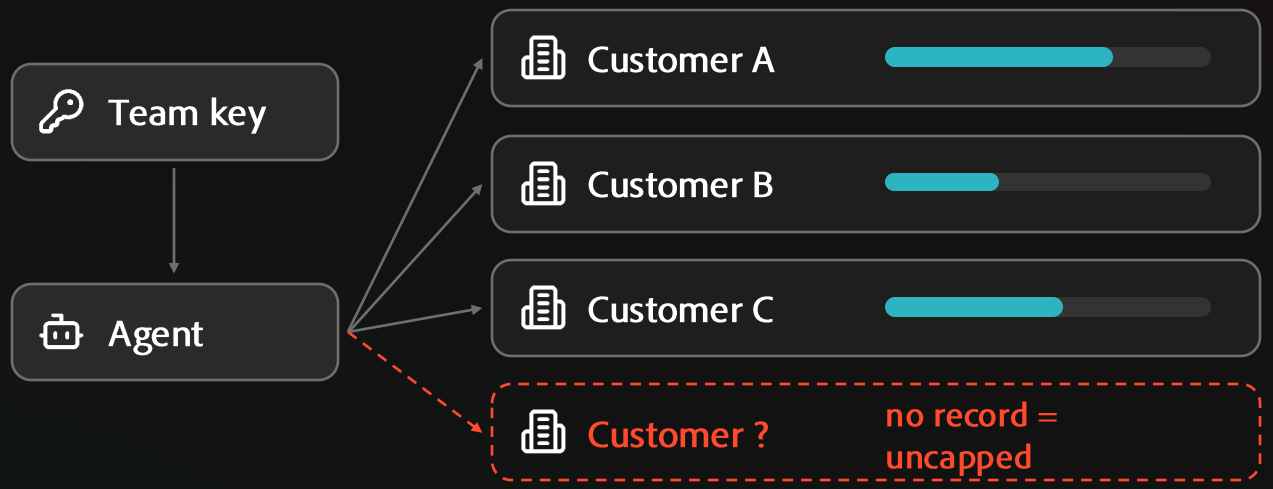
The dashboard showed a budget. Nothing enforced it.

THE FIX · FOLLOW THE OWNER

1. Stamp the owner on every key
2. Reconcile each member's budget onto one owner tag
3. Enforce the tag across all their keys

Customer-facing agents: the team isn't the spender.

WHAT HAPPENED



THE FIX · BUDGET THE CUSTOMER

1. The agent sends a customer ID on every request
2. The platform enforces a budget and rate limit per customer
3. Set a default budget per customer

The team decides who the customer is. The platform enforces the limit.

Every request is attributed

Attribution lives on the key — not in the request payload.

01 TAG THE KEY

ALWAYS PRESENT

user:<user-id>

OPTIONAL · ADD TO SELECTED KEYS

app:<app> env:<env>

→

02 USE THE KEY

EVERY REQUEST

Inherits the key's tags

NO PER-REQUEST TAGGING

No payload changes

→

03 ATTRIBUTE SPEND

ALWAYS ATTRIBUTED

By user

By app · environment

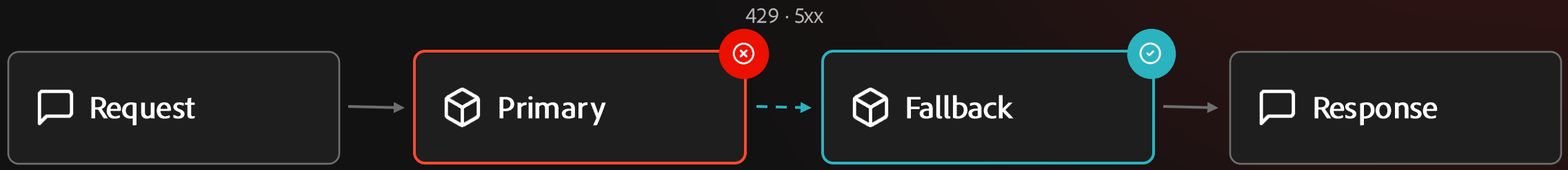
When tagged on the key

KEY-LEVEL SETUP Add custom tags to a key. Every request using it carries that context.

User attribution is automatic. Custom attribution is configured on the key.

LEVEL 3 · GUARANTEE 2 OF 3

Reliable and scalable. That's on us.



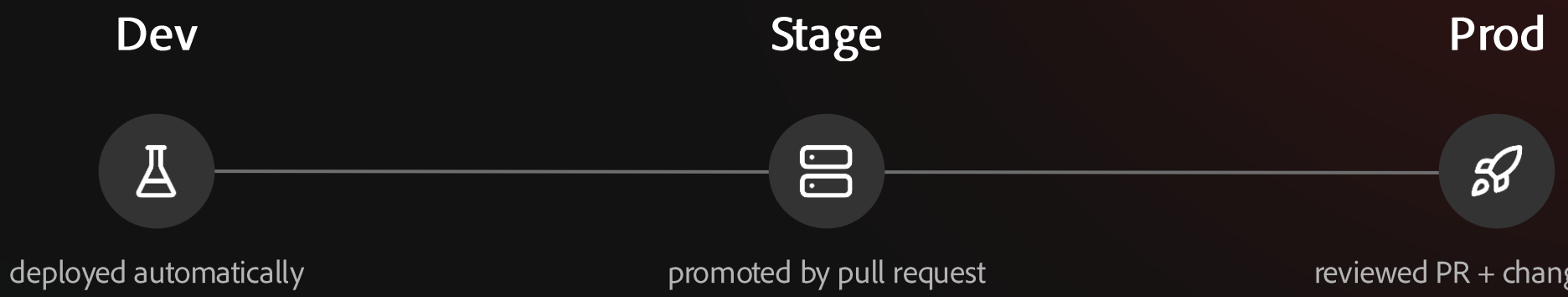
Fallbacks fire
Tested on every upgrade

The proxy scales
Autoscales on in-flight requests

Errors reach the tenant
Routed to the owning team

LEVEL 3 · GUARANTEE 3 OF 3

Upgrade on a train. Patch CVEs the same day.



Cooldown · new LiteLLM releases bake ~7 days before they move up

Contract suite · API contract tests gate every promotion

CVE fix · same day, skips the cooldown

Kubernetes made the guarantees boring.

Isolate the layers	A namespace per layer · NetworkPolicies between them
No keys in pods	Workload identity for platform, log-bucket and some model access
Stay up under agents	KEDA on in-flight requests · PodDisruptionBudgets · drain-safe grace period
Upgrade safely	Promotion is a GitOps pull request · Argo CD syncs and rolls back
Add a region	One cluster per region, same Helm chart · DNS in the values

WHAT'S NEXT

Next levels: region, cost, identity



IN PROGRESS

REGION

Multi-Region

Latency-based DNS sends callers to the nearest region. Residency means the whole stack moves.



PILOT

COST

Auto routing

A classifier routes to fast, reasoning or deep models. Evals decide if savings count.



PROTOTYPE

IDENTITY

Agent Identity

The owner travels with every request, instead of being stamped onto a key.

GAME SAVED

Achievements unlocked



LEVEL 1 ✓

Ownership follows the team



LEVEL 2 ✓

Budgets follow the owner, not the key



LEVEL 3 ✓

Guarantees are the platform's job

The proxy was simple.
The platform is the game.

Thank you

CONTINUE? ▶ Q&A

Dibyo Mukherjee

Tushar More

KubeCon | CloudNativeCon North America 2026

Adobe