IMESH

CASE STUDY | FINANCIAL SERVICES · INSURANCE | ANONYMISED

Securing a multi-cloud Istio mesh at insurance scale

How IMESH helped a leading public-sector general insurer in India adopt open-source Istio across 8 Kubernetes clusters — for zero-trust security, global rate limiting, and a Gateway API–compliant ingress that retires NGINX.

CLIENT
Public-sector general insurer (India)

ENGAGEMENT
2026 · OSS Istio adoption

SCOPE
Security · Rate limiting · Ingress

INDUSTRY General insurance · public sector	ANNUAL REVENUE Nearly $2B USD
CLOUD FOOTPRINT AWS (EKS) + Azure (AKS)	MESH Open-source Istio (no lock-in)
PRIMARY DRIVERS Zero-trust security · rate limiting	INGRESS Istio gateway via Gateway API

01 · THE CLIENT

A nationwide insurer modernising on Kubernetes

The client is one of India's largest public-sector general insurance providers, with nearly **$2B USD in annual revenue** and a policyholder base spanning the entire country. Their digital estate — policy issuance, claims, payments, partner integrations and customer portals — runs as a large microservices platform that had recently begun standardising on Amazon EKS as part of a broader move to cloud-native infrastructure.

As the EKS rollout matured, the platform team set out to adopt an **open-source Istio service mesh** to bring consistent security, traffic control and observability across every workload — without taking on proprietary lock-in. The scale involved made this a non-trivial undertaking.

8

Kubernetes clusters across AWS and Azure, covering every environment — dev, test, UAT, production and DR.

700+

Worker nodes under management, pooled across the multi-cloud cluster fleet.

1,000s

Microservices in production, each needing identity, encryption and traffic policy.

~5 lakh

Requests per second at peak load — renewal cycles, claims surges and campaign traffic.

02 · THE CHALLENGE

Three hard requirements, no margin for downtime

Running on EKS gave the team elastic compute, but it left the cross-cutting concerns — who can talk to whom, how traffic is throttled, and how it enters the platform — solved inconsistently across thousands of services. The team had a clear mandate but limited in-house mesh expertise, and the platform handles regulated financial workloads that cannot tolerate a disruptive cutover.

Zero-trust security across the mesh

Service-to-service traffic was largely unencrypted and authorised by network position. For a regulated insurer, the target was workload identity, **mTLS everywhere**, and fine-grained access policy — applied uniformly across 8 clusters and two clouds.

Rate limiting that holds at peak

With peaks near **5 lakh RPS**, the platform needed protection from traffic spikes, abusive clients and downstream overload — enforced consistently at the edge and between services, not bolted onto each application.

Replace NGINX with a Gateway API ingress

Ingress ran on NGINX with bespoke annotations and config drift across clusters. The team wanted to standardise on the **Istio ingress gateway driven by the Kubernetes Gateway API** — a portable, vendor-neutral spec — and retire NGINX without a risky big-bang switch.

03 · THE ENGAGEMENT

A phased, runbook-driven adoption

IMESH ran the programme as a sequence of low-risk phases, validating each in non-production before promoting to production. Every step was backed by runbooks, rollback paths and load testing — so the mesh was introduced under live traffic without disruption.

PHASE 01 · ASSESS & DESIGN
Discovery across all 8 clusters — traffic flows, security requirements and ingress inventory — leading to a target architecture for OSS Istio and a Gateway API ingress design.

PHASE 02 · MESH FOUNDATION
Hardened OSS Istio installed across AWS and Azure clusters with a consistent control-plane topology, plus observability wiring for metrics, traces and access logs.

PHASE 03 · SECURITY ROLLOUT
mTLS moved namespace-by-namespace from permissive to STRICT, with AuthorizationPolicies codifying least-privilege access — no application code changes required.

PHASE 04 · INGRESS MIGRATION

Istio ingress gateways stood up alongside NGINX using Gateway API resources, then traffic shifted by weighted cutover route-by-route until NGINX could be decommissioned.

PHASE 05 · RATE LIMITING & HANDOVER

Local and global rate limiting tuned and load-tested to peak, followed by knowledge transfer, runbooks and an upgrade cadence backed by 24×7 IMESH support.

04 · THE ARCHITECTURE

One mesh, two clouds, a single ingress model

All external traffic enters through Istio ingress gateways defined with the Gateway API, where TLS is terminated and global rate limiting is enforced. Inside each cluster, an Istio control plane manages Envoy sidecars that carry mTLS-encrypted, policy-governed traffic between services.

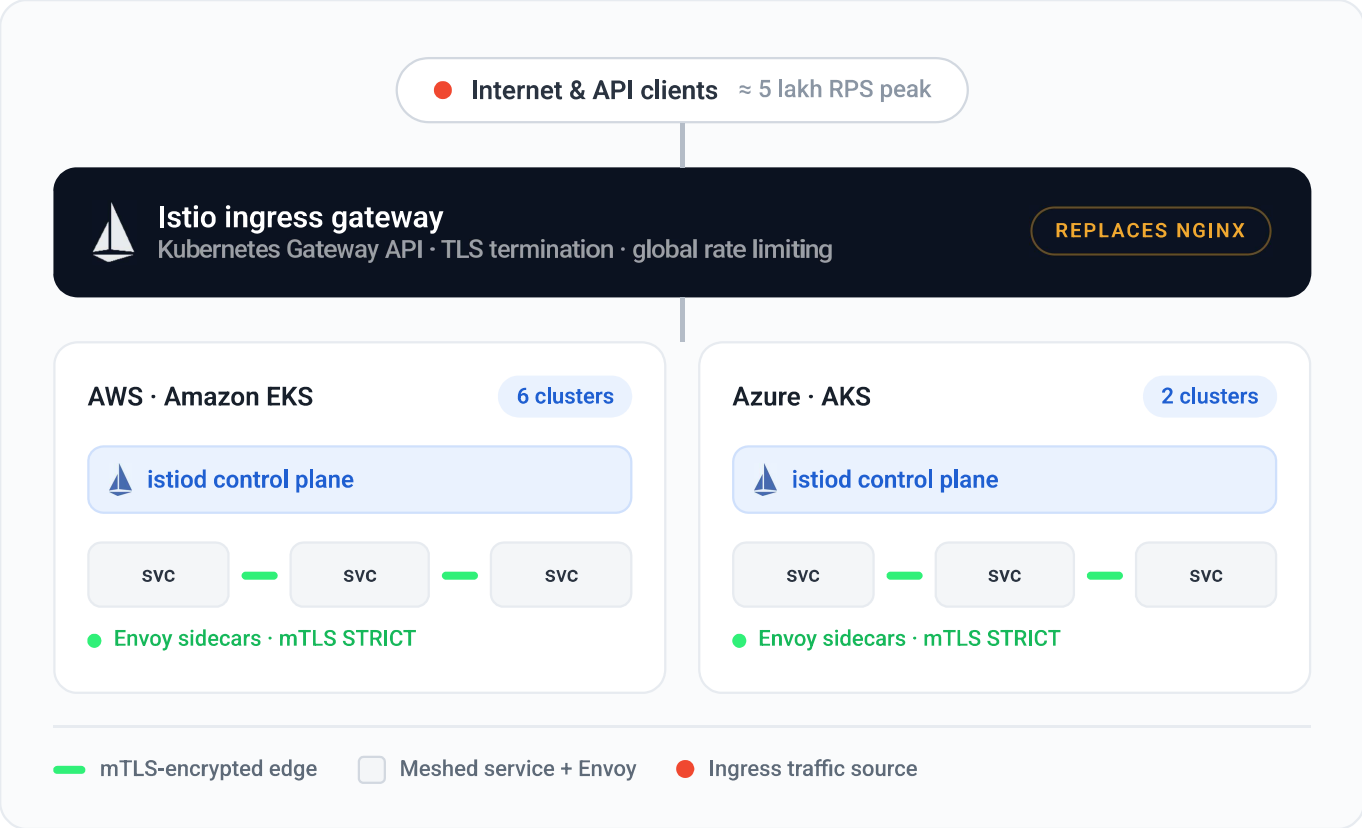

Figure 1 — Target architecture: a single Gateway API ingress model fronting OSS Istio meshes across 8 EKS/AKS clusters in AWS and Azure.

05 · THE SOLUTION

What IMESH delivered

Zero-trust security

Every workload received a cryptographic identity and all in-mesh traffic moved to **STRICT mTLS**, with certificate issuance and rotation handled by Istio. Access between services is governed by least-privilege `AuthorizationPolicy` rules rather than network position.

- Workload identity and mutual TLS across all 8 clusters.
- Namespace-scoped authorization, applied without app changes.
- JWT validation and request authentication at the edge.

Rate limiting at the edge and between services

A two-tier model combines fast **local rate limiting** in each Envoy proxy with a **global rate-limit service** backed by Redis for shared counters. Limits were load-tested to peak so the platform sheds abusive or runaway traffic before it reaches application tiers.

- Global limits enforced at the Gateway API ingress.
- Per-route and per-client quotas for sensitive APIs.
- Validated against ≈5 lakh RPS peak load.

Gateway API ingress, NGINX retired

Ingress was rebuilt on the Istio gateway using the **Kubernetes Gateway API** — a portable, role-oriented spec. Routes migrated off NGINX by weighted cutover, so the switch was gradual and reversible, ending with NGINX fully decommissioned.

- `Gateway` and `HTTPRoute` resources replacing NGINX annotations.
- Consistent, vendor-neutral config across every cluster.
- Zero-downtime, route-by-route cutover.

Outcomes

100%

mTLS coverage across in-mesh service traffic.

Zero

Downtime during the NGINX-to-Istio ingress cutover.

~5 lakh

RPS sustained at peak with rate limiting active.

8 / 8

Clusters brought under one consistent mesh model.

1

Portable Gateway API ingress standard, NGINX retired.

24×7

IMESH support, with upgrade cadence and runbooks.

The insurer now runs open-source Istio in production across two clouds — with zero-trust security, peak-grade rate limiting, and a single Gateway API ingress standard — and no proprietary lock-in.

Delivered by IMESH · 2026

This case study is anonymised at the client's request. Scale and performance figures are representative of the engagement and can be finalised against the client's production telemetry.