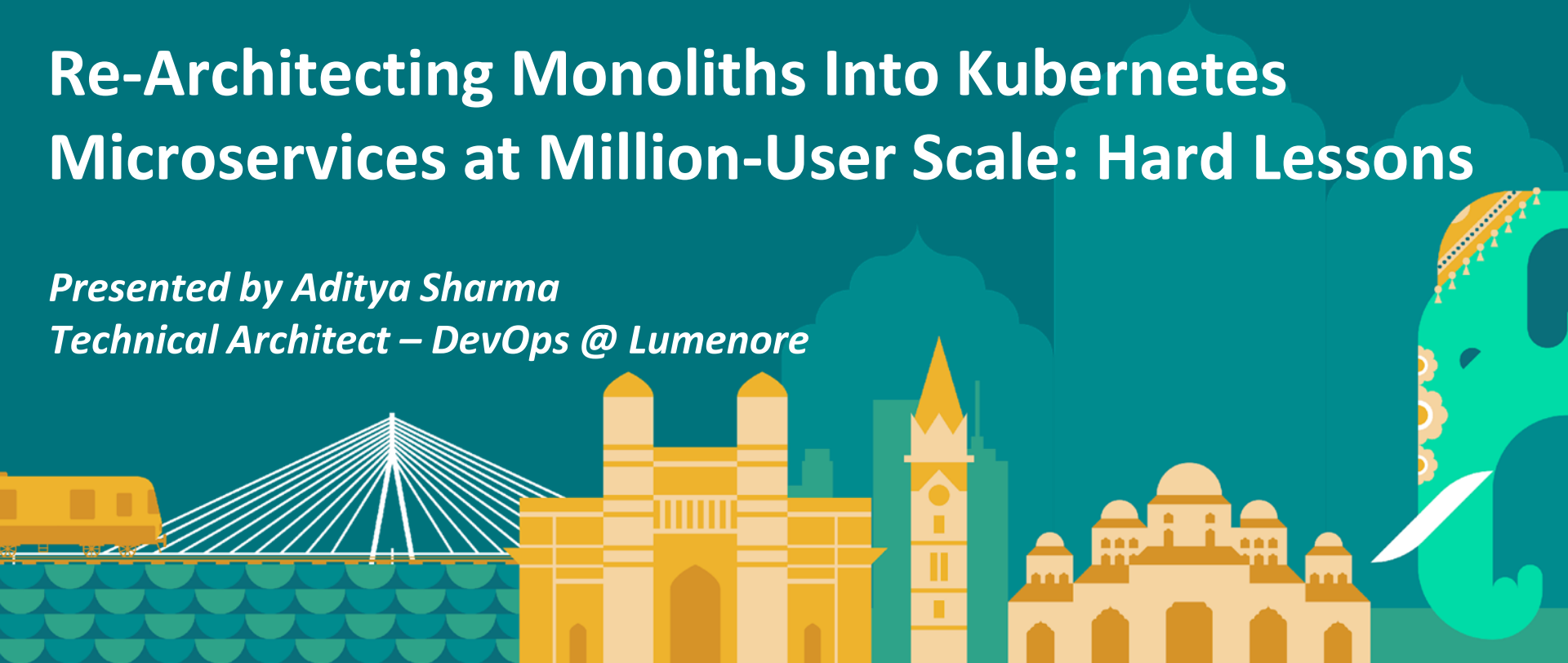
KubeCon CloudNativeCon India 2026

#KubeCon #CloudNativeCon

Re-Architecting Monoliths Into Kubernetes Microservices at Million-User Scale: Hard Lessons

Presented by Aditya Sharma
Technical Architect – DevOps @ Lumenore



Setting the Stage: One Platform, Many Hard Lessons

KubeCon CloudNativeCon India 2026

What You're About to Hear

- A 10+ year saga of building **Lumenore - a Business Intelligence & AI based Data Analytics platform** - from bare metal to cloud-native
- What Broke, What Hurt and What We Built to Make Sure It Never Happened Again
- How we transformed platform to handle millions of users across cloud and on-premises

About The Speaker

kubestronaut

Aditya Sharma

Technical Architect - DevOps @ Lumenore

- 8+ years of experience in DevOps and Software Engineering - building, breaking, and fixing at scale
- Kubestronaut Certified, and Golden Kubestronaut on the way
- Armed with 25+ CNCF, Cloud, and IT certifications

KubeCon CloudNativeCon India 2026

Chapter 1: Where It All Began

Year 0-2 The Monolith Era - One Big App to Rule Them All

KubeCon CloudNativeCon India 2026

4 Servers Total

2 Application

2 Database

CentOS on Bare-Metal Servers

App Server 1

Java

Application Servers

App Servers 2

Polymer JS

DB Server 1

MySQL (Transactional DB)

Database Servers

DB Servers 2

Vertica (Data Analytical DB)

TortoiseSVN
Code & Data pulled by Application Servers

First Pain Points

- HTTP 5xx as user count grew past 1000
- CPU throttling + memory spikes under load
- No rolling updates - every fix needed downtime

The Hard Lessons

- Vertical scaling helped temporarily, Cost vs ROI turned negative fast.
- Every change required downtime.
- Vertical and horizontal scaling has a hard ceiling and a slow, expensive approval process
- Hardware upgrades during maintenance windows = frustrated customers

Key Insight

Availability and scalability must be designed in - they cannot be bolted on later

Year 3-4 The Great Unbundling - Enter Docker and Microservices

KubeCon CloudNativeCon India 2026

App Server 1

Java

100+ Replicas

App Servers 2

Polymer JS

100+ Replicas

Application Servers (Docker creates App Replicas)

DB Server 1

MySQL (Transactional DB)

DB Servers 2

Vertica (Data Analytical DB)

Database Servers

Observability Stack (First Generation)

Zipkin
Distributed tracing.

Docker Stats
CPU / Memory / Network metrics

Logs to NFS
Logs shipped to NFS-mounted storage

GlusterFS
Scalable Network Filesystem to share the data storage between the servers.

TortoiseSVN → GitLab → Jenkins

The Containerization Shift

- 6-8-month POC → Docker was installed in all the servers.
- Replica-based scaling replaced running application directly over VM.
- Costs dropped significantly as we required less hardware to run the same application.

What the Data Revealed

- Monolithic codebase becoming unmanageable - thousands of lines, impossible to debug
- Java and Polymer JS was not sufficient to create all features we had in our Roadmap.

The Decision

- Decompose the monolith - well-defined service interfaces, clear ownership → Microservice was the future.
- The codebase was branched into different languages and frameworks

KubeCon CloudNativeCon India 2026

Chapter 2: Kubernetes Enters the Chat - The Orchestration Era Begins

Year 5 Kubernetes Arrives - Power, Promise and a Painful Outage

KubeCon CloudNativeCon India 2026

DNS resolved to Master Node IP

Kubernetes Cluster
(3 Servers Total)
1. Master Node
2. Worker Nodes

GitLab (Git running on VM)
Code & Data pulled by Application Servers

MASTER NODE
kube-apiserver
scheduler
controller-manager

WORKER NODE 1
kubelet
Kube-proxy
docker

WORKER NODE 2
kubelet
Kube-proxy
docker

Java | Python | Spark (Big Data) | Node.js (UI) | Redis (Cache) | MongoDB (Database) | ... More Services

Application Stack (Running as Pods)

The Outage - And What It Forced

- A single Kubernetes node failure → complete platform outage
- Root cause: single control plane, no node redundancy designed for failure.
- Monitoring alerts or stats being missed if pod or nodes goes down.

Database Servers (External to K8s)

DB Server 1
MySQL (Metadata Transactional DB

DB Servers 2
Vertica (Data Analytical DB)

Observability Stack v2

Prometheus (Metrics)

Grafana (Dashboards & Alerts)

ELK Stack (Logs)
Elasticsearch, Logstash, Kibana

Zipkin (Traces)

The Response

Decided to plan and switched to a highly available architecture which would provide resilience and zero-downtime operations

Year 6 Never Again! - Our Crash Course in High Availability

KubeCon CloudNativeCon India 2026

DNS resolved to HAProxy Keepalived IP

HAProxy 1 (Keepalived) ⟷ HAProxy 2 (Keepalived)

GitLab (Git running on VM)
Code & Data pulled by Application Servers

Control Panel (3 Master Nodes)

Master 1	Master 2	Master 3
kube-apiserver controller-manager Scheduler, etcd	kube-apiserver controller-manager Scheduler, etcd	kube-apiserver controller-manager Scheduler, etcd

Application Panel (3 Worker Nodes)

Worker 1	Worker 2	Worker 3
kubelet Kube-proxy	kubelet Kube-proxy	kubelet Kube-proxy

Observability Stack v3 (HA)

Prometheus (HA) ⟷ **Grafana** (HA) ⟷ **Alert Manager** (HA)

Upgraded monitoring with Highly available mode of Prometheus. Grafana, Alertmanager using kube-prometheus-stack deployed with Helm chart.

The HA Architecture Response

- Multi-node control plane with etcd quorum
- Worker node redundancy with zone-aware scheduling
- Node failures became non-events: K8s version upgrades or any OS maintenance now zero-downtime

Key Limitations of This Setup

- This setup was regional and stuck to one location.
- It was very costly to change the CPU, RAM, storage type, speed, or vendor.
- It was difficult to manage all the hardware and software with growing environment count, code bases, and the user base.

Chapter 3: Going Global - Multi-Region, Multi-Cluster, Multi-Everything

Year 7-8 Run Anywhere, Break Nowhere - Building a Truly Portable Platform

KubeCon CloudNativeCon India 2026

AZURE/ORACLE/AWS-Elastic Services

Kubernetes Cluster

Node 1 | Node 2 | Node ...N → New Node Added

DC-DR Setup

East US (DC) ⟷ Replication & Sync ⟷ West US (DR)

Database for MySQL and Redis

- Fully managed MySQL (later changed to PostgreSQL), and Redis.
- Automated backups.
- High availability.

Storage account & Key Vault

- Fully managed Data Storage and credentials storage.
- High availability with DR.
- Encryption and data backups.

Azure DevOps and Container Registry

- Code was versioned controlled in Azure Repos.
- Docker Images were stored in ACR.

Lumenore was packaged with Helm charts and Kustomize.

Terraform
Infrastructure as Code for provisioning and managing environments.

Ansible
Automation for configuration management and deployment.

With Helm charts, Kustomize, Terraform, and Ansible, Lumenore can be deployed in any cloud, including Oracle, AWS, Azure, Google, etc., and on-premise environments.

ORACLE aws On Premise

KEY OUTCOMES

- Scalable & Elastic Infrastructure
- High Availability & Resilience
- Cost Optimized
- Faster Delivery with Automation
- Security & Compliance
- Business Continuity with DC-DR

CILIUM CLUSTER MESH

cilium

Microsoft Azure (AKS): East US Cluster ⟷ West US Cluster

Cilium Cluster Mesh

Amazon Web Service (EKS): East US Cluster ⟷ West US Cluster

Cilium Cluster Mesh

Cilium Cluster Mesh connected multiple clusters across regions/clouds for resource and load distribution to provide cost effective proper or all resource utilized.

ON-PREM SETUP enhanced

containerd | Cilium | MetalLB | Ingress Nginx | Rook Ceph

- On-prem setup was done using containerd instead of Docker.
- Ingress-Nginx was later deprecated and Gateway API was used.
- Lumenore can be run in any OS where K8s can run like RHEL, AlmaLinux, Ubuntu, Amazon Linux, etc.

KubeCon CloudNativeCon India 2026

Chapter 4: Security Gets Real - From Assumed to Actually Engineered

Year 9-10 Locking It All Down - Security Across the Code, Container, Cluster & Cloud.

KubeCon | CloudNativeCon India 2026

1 CODE & CONTAINER REMEDIATION

SonarQube & OWASP Dependency Check
Static analysis of code quality and open source dependencies integrated into CI pipeline

Trivy
CVE, exposed credentials, license scanning on all container images, code repositories and even VMs

Docker Multi-Stage Build
Builds Smaller images, reduced attack surface and improved security

Syft SBOM and Cosign Image Signing.
Creates a complete inventory of software components and cryptographically signs images to ensure integrity and trust.

KubeLinter & Hadolint
Static analysis of Kubernetes manifests and Dockerfiles to enforce best practices and standards

CI / CD PIPELINE INTEGRATION
Commit → Build → Scan → Quality Cate → Deploy

2 CLUSTER & CLOUD REMEDIATION

Pod Security Admission (Formerly Policies)
Restricted profile enforced across all namespaces to prevent privilege escalation

Kubernetes Network Policies
Least-privilege inter-pod communication - deny everything by default with explicit allow rules

Falco, Kyverno, Kubescape & Trivy Operator
Real-time threat detection, policy enforcement and vulnerability scanning for compliance and runtime security

Falco | Kyverno | Kubescape | trivy

Cilium Hubble UI
Deep visibility into network flows, monitor traffic, debug network policies and identify dropped packets

Workload → Network Policy → Hubble UI

3 CLOUD LAYER HARDENING

Network Security Group (NSG)
NSG rules tightened to explicit allow-lists; unnecessary ports closed and traffic minimized

Identity & Access Management
Least-privileged access, MFA, RBAC, KMS, encryption at rest and in transit implemented. Secret Management with rotation policies.

RBAC

Backup & DR and Patch Management
Encrypted off-site backups, RTO/RPO-defined recovery plans. OS and node image auto-patching, vulnerability SLAs

Audit Logging & SIEM
Comprehensive audit logging enabled across Kubernetes clusters and all other cloud services and integrated with Wazuh SIEM for monitoring. detection and alerting

WAZUH

Edge & Application Protection
WAF, Firewall and DDoS Protection enabled for every endpoint with maximum security parameters

KubeCon CloudNativeCon India 2026

Chapter 5: The Final Boss - From Good Enough to Truly Production Grade

Year 11-13 The Dream Stack - Resilient, Observable, Automated

KubeCon CloudNativeCon India 2026

1 ARGO CD - GITOPS AS THE SOLUTION

What Was Happening

- Engineers made direct changes via Kubernetes dashboard - skipping Git
- Jenkins pipelines later overwrote those changes silently
- No single source of truth - live cluster state and repository diverged constantly

Argo CD - GitOps as the Solution

- All manifests (Deployments, Services, ConfigMaps, NetworkPolicies, HPAs) moved to Git
- Argo CD continuously reconciles live state against declared state
- Manual drift detected and corrected automatically. and Rollbacks became Git reverts - fast. traceable. low-risk
- Full audit trail: who changed what, when, and why

2 ISTIO – Service Mesh

Why Istio Was Needed

- mTLS between services required code changes in every service - impractical
- Transient failures cascaded across dependent services
- Reroute traffic to healthy endpoint or A/B deployment was missing

Istio Capabilities Adopted

Strict mTLS - mutual TLS between all services. transparent to application code

Retries - transient failures absorbed automatically,

Circuit Breaking - struggling services isolated; downstream load shed gracefully

Traffic management - canary deployments. weighted splits, real-time rerouting

Multi-Cluster Service Mesh - Securely connect and manage services across multiple Kubernetes clusters.

Zero Trust Networking - Enforce service identity, authentication, and authorization for every communication.

Using Kiali UI - Real-time service graph - traffic flows, error rates, latency per service

3 OPENTELEMETRY – OBSERVABILITY

Why OpenTelemetry Was Needed

- Every service had its own logging and tracing setup - inconsistent, hard to correlate.
- No unified signal - metries, traces, and logs lived in separate silos
- To get metries or traces from code, application code needs to be modified.

OpenTelemetry: What It Solved

- Auto-instrumentation across all services - zero code changes per application.
- Unified observability all three signal types.
- Vendor-neutral - backend can change without touching application code.

Visualized in Grafana

- Metrics, Traces, Logs - in one place
- Better correlation and faster troubleshooting
- Standardized observability across all clusters (Cloud & On-Prem)

Thanks for riding along! We hope our war stories save you a few scars.

Let Us Stay Connected

LinkedIn: https://www.linkedin.com/in/adityasharmadevops
Email: adityas@lumenore.com
Website: https://lumenore.com