

Architettura Microservizi Event-Driven su Kubernetes

Implementazione di una pipeline IoT scalabile per la gestione differenziata di flussi ad alta frequenza e criticità.

Studente: Enea Manzi

Corso: Cloud Computing Technologies

Anno Accademico: 2025/2026

| Obiettivi Architetturali & Requisiti

└ Scenario: Monitoraggio real-time di una rete eterogenea di sensori IoT industriali.

Project 1: Apache Kafka

Sistema di messaging asincrono

- ✓ **Fault Tolerance & HA**
- ✓ **Advanced Security** (*channel encryption, authentication*)
- ✓ **Topic Differenziati per QoS**

Project 3: Kong API Gateway

Edge API Gateway and Ingress Controller

- ✓ **Gateway Offloading:** Autenticazione (*API Key*) e Load Balancing
- ✓ **Security:** Rate Limiting dos protection
- ✓ **Infrastructure as Code:** Kubernetes CRD

Architettura e Flusso Dati (Event-Driven)

La pipeline è segmentata in namespace logici ('kong', 'kafka', 'metrics') per garantire la Separation of Concerns.



- **Kong GW:** Gateway Offloading & API Key Validation.
- **Producer:** Data Enrichment (UUID) & Smart Routing.
- **Kafka:** Buffer Persistente & Backpressure Management.
- **Consumer:** Normalizzazione & Zero Data Loss Logic.
- **MongoDB:** Time Series Storage (Zstd Compression).
- **Metrics:** Analytics & Aggregazioni On-Demand.

Apache Kafka: Topic Strategy & Durability

KRaft Mode (No ZooKeeper)

✓ Metadata Management Interno Controller & Broker NodePools separati Risorse minori

Topic: sensor-telemetry (Alta Frequenza)

Flussi massivi (Telemetria, Boot, Update): **Efficienza**

```
partitions: 3  
config:  
  compression.type: lz4  
  retention.ms: 604800000
```

Topic: sensor-alerts (Critico / Durabilità)

Eventi sporadici (Allarmi): **Garanzia di Consegnna**

```
partitions: 2  
config:  
  min.insync.replicas: 2  
  retention.ms: 2592000000
```

compression ibrida

Kafka: LZ4

Transport Layer
Bassa latenza

MongoDB: Zstd

Storage Layer
Disk efficiency

Kong Gateway: Edge Computing & Security Offloading

🔒 Authentication

GATEWAY OFFLOADING

Plugin

key-auth

Check vs

Kubernetes Secrets

Block

401 Unauthorized (Edge)

Zero-Downtime Revocation

🚦 Traffic Control

PROTECTION & ROUTING

Policy: Rate Limiting

5 req/sec

Defense

Anti-DoS / Flood (429)

Routing: Load Balancing

Round-Robin

Gestione nativa:

Service + Kong Ingress

⚙️ Configuration

INFRASTRUCTURE AS CODE

Format

Kubernetes CRD

Approach

100% Declarative

Ops

No UI / CI/CD Ready

kind: Ingress
annotations:
 konghq.com/plugins: key-auth, global-rate-limit

| Persistenza: MongoDB Time Series Collection

Time Series & Storage

Compressione Zstd

Risparmio disco ~70%

Indici Clustered

Query veloci su range temporali

TTL Automatico

Pulizia dati dopo 30 giorni

Infrastruttura (StatefulSet)

StatefulSet vs Deployment

Identità stabile

PVC Binding

Riacquisizione disco garantita dopo crash

Identità di Rete

mongo-0 persistente

Configurazione (Cloud Native)

ConfigMap

Host, Port, DB Name (Non sensibili)

Secrets

User/Password (Iniettate come Env Vars)

No Hardcoding

Best Practice di sicurezza

Microservices Implementation

Tutti i servizi sono **containerizzati** (Python 3.11) e gestiti come Deployment/StatefulSet su Kubernetes.

PRODUCER

Stateless API
(Flask)

Enrichment
UUID + Timestamp

Pattern
Idempotency Token

Sec
SASL/TLS 9093

CONSUMER

Stateful Worker

Action
Normalize & Store

Feature
Buffering (Zero Data Loss)

Scale
3 Replicas (match Partitions)

METRICS

Analytics Service

Source
MongoDB Time Series

Endpoints
5 Aggregations

Query
Real-time on Indexed Data

| Non-Functional Properties: Validation Matrix

Le NFP sono validate tramite scenari di test specifici (Demo Live).

SECURITY

Data in Transit

TLS 9093 + SASL/SCRAM

Edge Protection

API Key & Rate Limit

Secrets

K8s Native Management

RESILIENCE & HA

Fault Tolerance

Kafka Buffering (Zero Data Loss)

High Availability

Self-Healing & Auto-Recovery

SCALABILITY

Load Balancing

Round-Robin

Parallelism

3 Replicas / 3 Partitions

Auto-Scaling

HPA (CPU > 50%)

Sfide Tecniche & Soluzioni Consapevoli

NETWORKING & REACHABILITY

Problema:

Docker Desktop Network Isolation blocca l'accesso agli Ingress.

Soluzione:

Migrazione a Minikube Native Driver per esporre IP cluster.

Risultato:

Esposizione diretta e risoluzione DNS nip.io funzionante.

IDEMPOTENCY & AT-LEAST-ONCE

Problema:

Rischio di messaggi duplicati se il Consumer crasha pre-commit.

Soluzione:

Implementazione UUID Univoco nel Producer.

Risultato:

Architettura pronta per Upsert Idempotente su DB.

STATEFULNESS MONGODB

Problema:

Deployment stateless perde il binding PVC al riavvio.

Soluzione:

Adozione del pattern StatefulSet per MongoDB.

Risultato:

Identità stabile (mongo-0) e riaggancio immediato disco.

Roadmap Demo



Sequenza di Validazione (20 min)

PARTE 1: Test Base

- Auth Check (401 vs 200)
- Pipeline completa: Invio eventi + Lettura Metriche

PARTE 2: NFP Validation

1. **Security:** TLS Verification + SASL + API key
2. **Fault Tolerance:** Consumer Crash & Recovery (Buffering)
3. **HA:** Pod Kill & Self-Healing
4. **Scalability:** Manual Scale Out & LB
5. **Elasticità:** HPA Stress Test
6. **Rate Limiting:** DoS Protection (429)



Q & A

Grazie per l'attenzione. Sono disponibile per domande sull'architettura e l'implementazione.

github.com/eneamanzi/k8s-kafka-kong