HAZELCAST

Internal Technical Brief

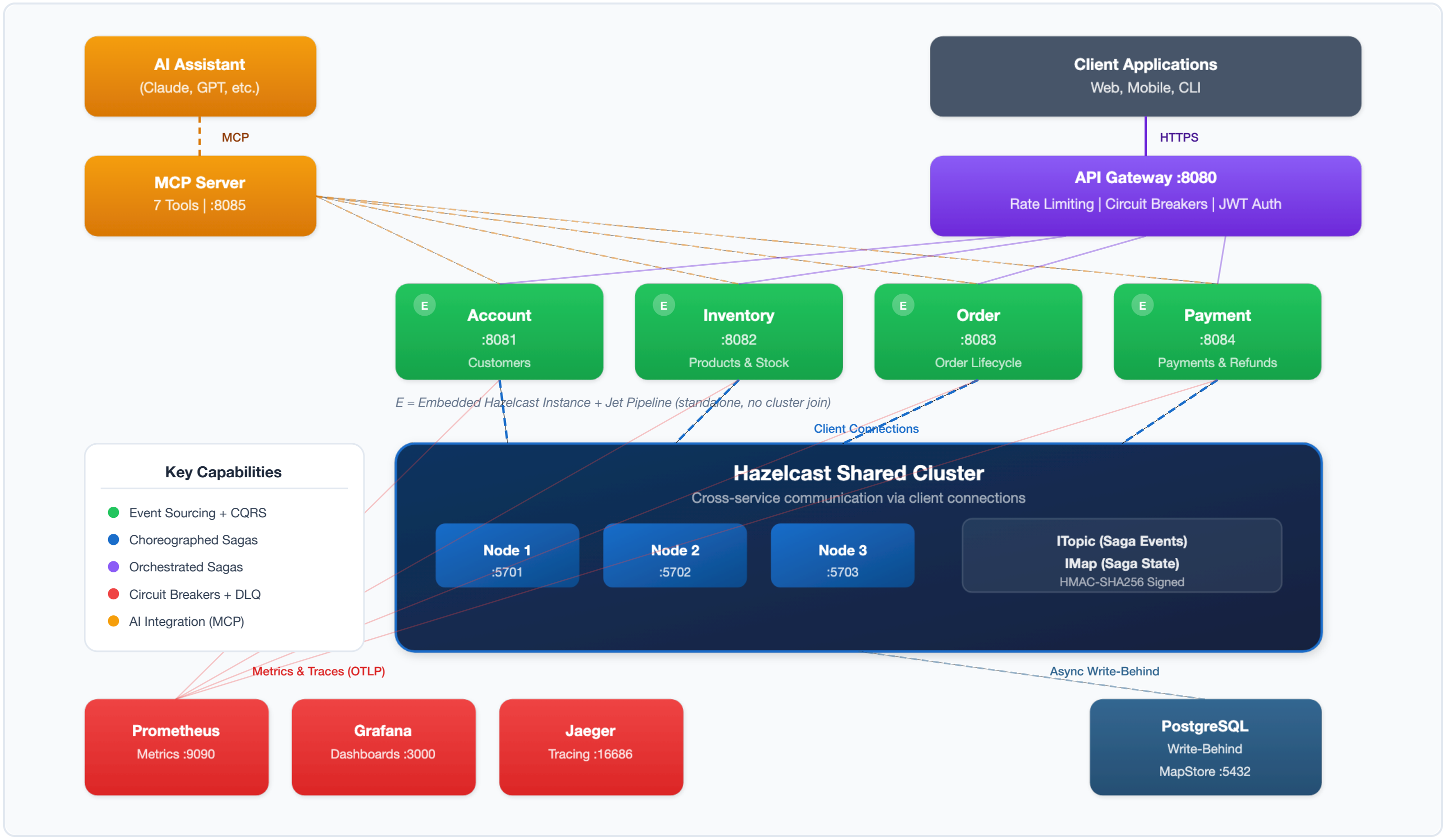
Microservices Framework

Production-ready event sourcing, saga coordination, and real-time stream processing — built entirely on Hazelcast and Spring Boot.

100K+	<1ms	54K+	12.6ms	7	4
VIEW OPS/SEC	P99 VIEW LATENCY	SAGAS EXECUTED	P50 E2E LATENCY	MCP AI TOOLS	MICROSERVICES

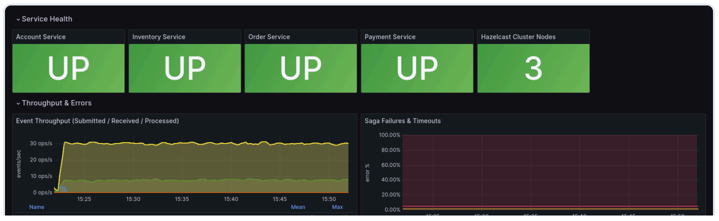
The gap: Customers evaluating Hazelcast for microservices ask "show me event sourcing," "show me sagas," "show me how Jet fits." We point them at docs and samples. Competitors ship turnkey frameworks with dashboards and demos. **This framework closes that gap** — a complete, instrumented, four-service eCommerce system that exercises every Hazelcast distributed-systems primitive end-to-end: IMap event stores, Jet pipelines, ITopic event bus, saga coordination, circuit breakers, write-behind persistence, and AI integration via MCP — all with pre-provisioned Grafana dashboards that make the value visible in real time.

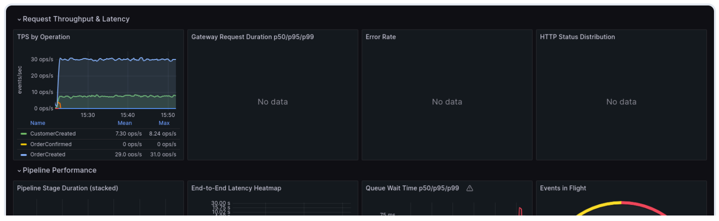
System Architecture



Dual-instance architecture: each service runs an embedded standalone Hazelcast (Jet pipelines) + client connection to shared 3-node cluster (cross-service sagas via ITopic)

Live Observability — Pre-Provisioned Dashboards




Request Throughput & Latency
TPS by Operation
Gateway Request Duration p50/p95/p99
No data
Error Rate
No data
HTTP Status Distribution
No data
Pipeline Performance
Pipeline Stage Duration (stacked)
End-to-End Latency Heatmap
Queue Wait Time p50/p95/p99
Events in Flight

Name	Mean	Max
CustomerCreated	7.30 ops/s	8.24 ops/s
OrderConfirmed	0 ops/s	0 ops/s
OrderCreated	29.0 ops/s	31.0 ops/s

● Event Sourcing & CQRS
Append-only event store backed by IMap with event journal. Materialized views rebuilt from events via Jet pipelines deliver 100K+ ops/sec at sub-millisecond latency.

● Distributed Sagas
Both choreographed (event-driven, no coordinator) and orchestrated (central controller, HTTP steps) saga patterns with automatic compensation, timeout detection, and per-step metrics.

● Resilience & Reliability
Resilience4j circuit breakers with retry on all saga listeners. Transactional outbox for guaranteed delivery. Dead letter queue with admin REST API. Idempotency guards for exactly-once processing.

● API Gateway
Spring Cloud Gateway with path-based routing, Hazelcast-backed rate limiting (shared state across instances), per-route circuit breakers, JWT auth, CORS, and correlation ID propagation.

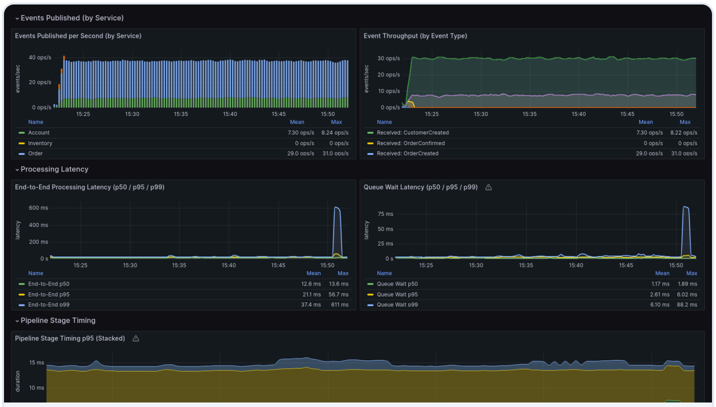
● AI Integration (MCP)
7 MCP tools let AI assistants query views, submit events, inspect saga state, check metrics, and run demos. Supports stdio and HTTP/SSE transport with API key RBAC.

● Full Observability
Micrometer + Prometheus metrics across all services. Six pre-provisioned Grafana dashboards. Jaeger distributed tracing via OTLP. Per-step saga duration metrics.

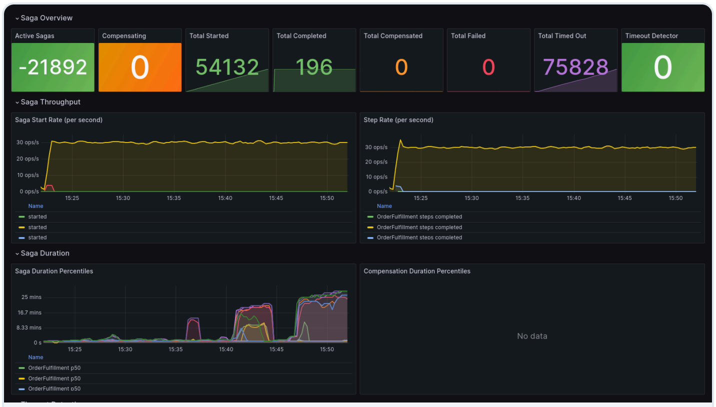
● Durable Persistence
Write-behind MapStore batches events and views to PostgreSQL asynchronously. Automatic cold-start reload. Bounded IMap memory with eviction. Provider-agnostic interfaces for any JDBC database.

● Security (3 Layers)
JWT/OAuth2 on gateway & services. HMAC-SHA256 signing of ITopic saga events for service-to-service authentication. MCP API key RBAC (VIEWER / OPERATOR / ADMIN).

Grafana Dashboard Suite


Events Published (by Service)
Events Published per Second (by Service)
Event Throughput (by Event Type)
Processing Latency
End-to-End Processing Latency (p50 / p95 / p99)
Queue Wait Latency (p50 / p95 / p99)
Pipeline Stage Timing
Pipeline Stage Timing p95 (Stacked)

Event Flow — Throughput, latency percentiles, pipeline stage timing



Saga Coordination — 54K sagas, compensation tracking, duration percentiles

Technology Stack

Component	Technology
Runtime	Java 17+
Framework	Spring Boot 3.2
Data Grid	Hazelcast 5.6 (CE default, EE opt-in)
Stream Processing	Hazelcast Jet
Database	PostgreSQL 16+ (write-behind)
Gateway	Spring Cloud Gateway
AI Integration	Spring AI MCP Server 1.0

Modules (9 Total)

framework-core	Domain-agnostic event sourcing engine
framework-postgres	Write-behind MapStore persistence
ecommerce-common	Shared domain objects, events, DTOs
account-service	Customer management (:8081)
inventory-service	Product catalog & stock (:8082)
order-service	Order lifecycle (:8083)
payment-service	Payments & refunds (:8084)
api-gateway	Routing, rate limiting, auth (:8080)

Get Started

Clone, build, and run the full four-service demo in under 5 minutes with Docker Compose. Pre-loaded sample data, demo scripts, and dashboards are ready out of the box. The framework is designed to be forked and extended for customer-specific demos and proof-of-concept engagements.

Repository: github.com/theyawns/hazelcast-microservices-framework
Quick Start: ./scripts/start-docker.sh
Demo: ./scripts/demo-scenarios.sh
Contact: mike.yawn@hazelcast.com