HAZELCAST

SA Technical Brief

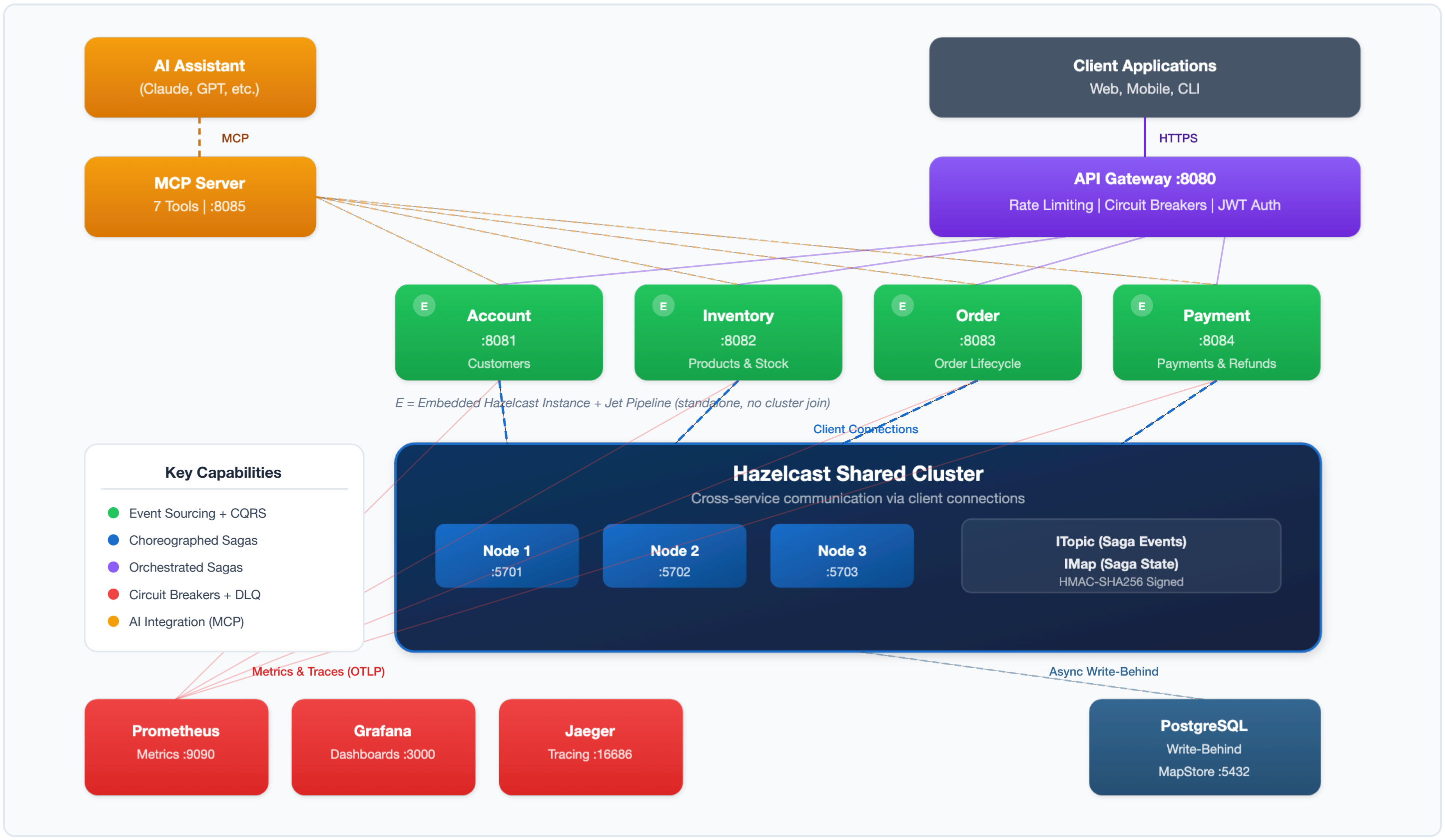
Microservices Framework

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

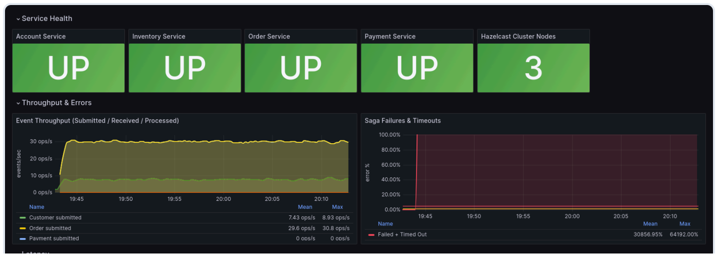
Hazelcast is a great platform for supporting microservice architectures. But for customers who aren't fully there yet, transitioning from a monolith to microservices is more than just breaking things apart — do it wrong and you end up with many tightly-coupled pieces where any failure breaks the whole system. Using microservice patterns effectively is the key. This demo and the accompanying series of blog posts walks prospects or customers through the right way to implement a Hazelcast-centric microservice architecture.

System Architecture



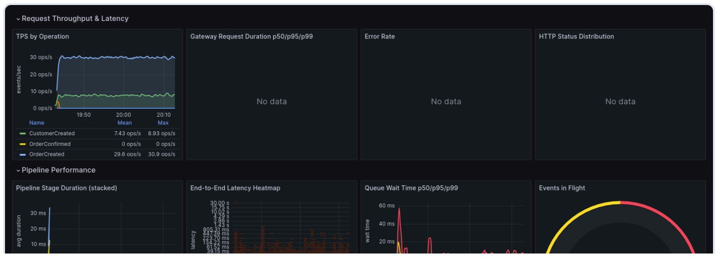
Dual-instance architecture: each service runs an embedded standalone Hazelcast (Jet pipelines) + client connection to shared 3-node cluster (cross-service sagas via ITopic)

Live Observability — Pre-Provisioned Dashboards



```
Name                  Mean        Max
Customer submitted    7.43 ops/s  8.93 ops/s
Order submitted       29.6 ops/s  30.8 ops/s
Payment submitted     0 ops/s     0 ops/s
```

```
Name               Mean       Max
Failed + Timed Out 30856.95%  64192.00%
```


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

```
Name            Mean        Max
CustomerCreated 7.43 ops/s  8.93 ops/s
OrderConfirmed  0 ops/s     0 ops/s
OrderCreated    29.6 ops/s  30.9 ops/s
```

Capabilities & Technical Details

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.

AI-Powered Similarity Search

Hazelcast VectorCollection with HNSW indexing for product similarity. Supports cosine, dot-product, and euclidean metrics. `GET /api/products/{id}/similar`. Enterprise opt-in with Community NoOp fallback.

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

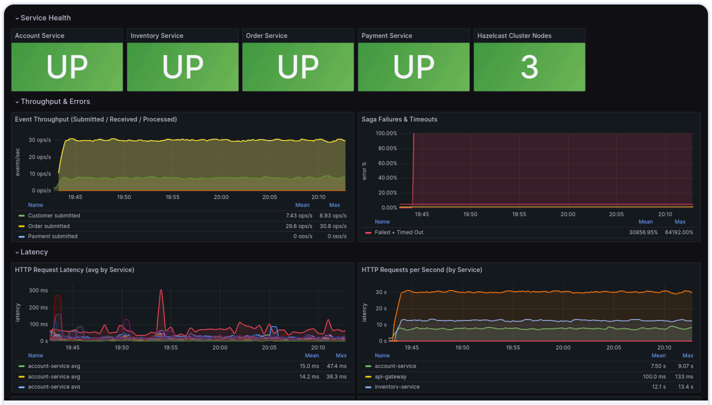
Write-behind MapStore batches events and views to PostgreSQL asynchronously. Automatic cold-start reload. Bounded IMap memory with eviction. Provider-agnostic interfaces for any JDBC database.

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).

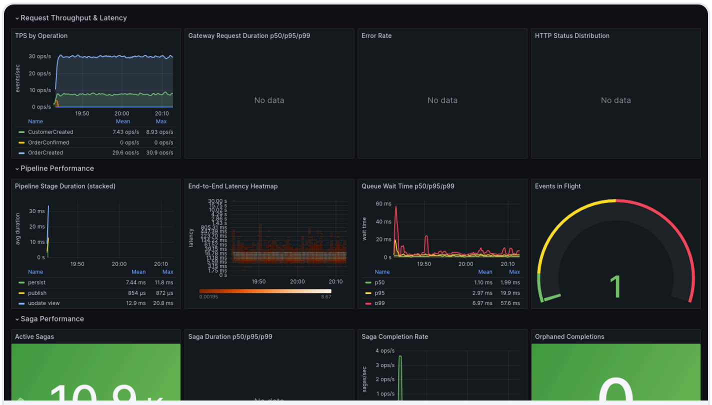
Deployment: Runs on a developer laptop via Docker Compose or deploys to AWS EKS with provided Helm charts and Terraform scripts. Other cloud providers (GCP GKE, Azure AKS) can be targeted by request.

Six pre-provisioned Grafana dashboards provide real-time visibility into every layer of the framework — from high-level system health down to individual event flow, saga coordination, materialized view performance, and database persistence. Dashboards are auto-provisioned via Docker Compose and Helm charts — no manual setup required.



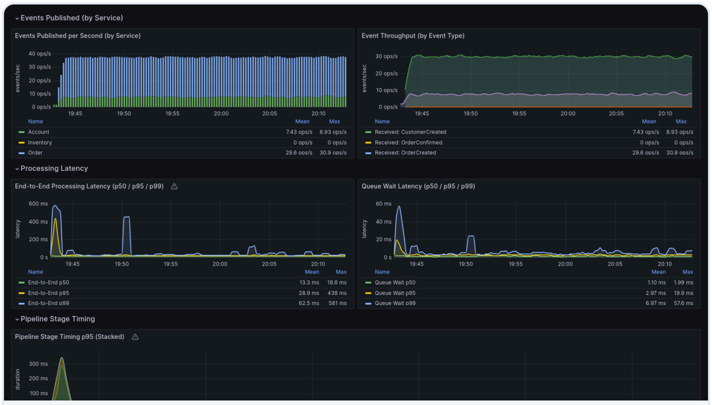
System Overview

Service health, JVM metrics, throughput, latency percentiles, error rates



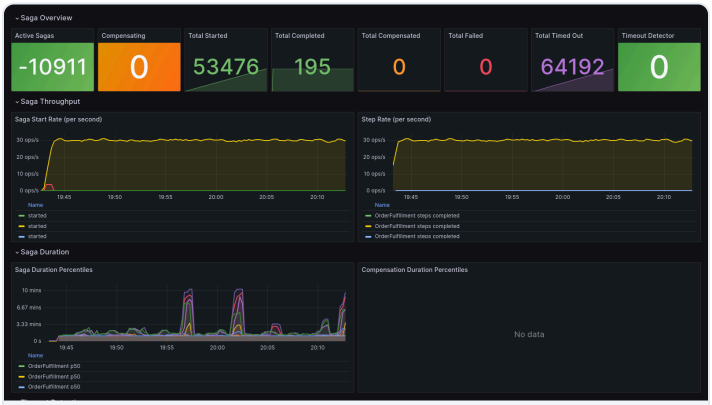
Performance Testing

TPS targets, pipeline throughput, saga completion rates, resource utilization


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)

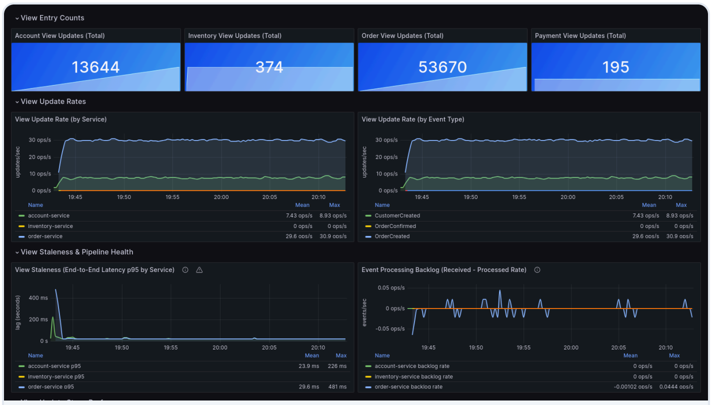
Event Flow

Event throughput, latency percentiles, pipeline stage timing, event types



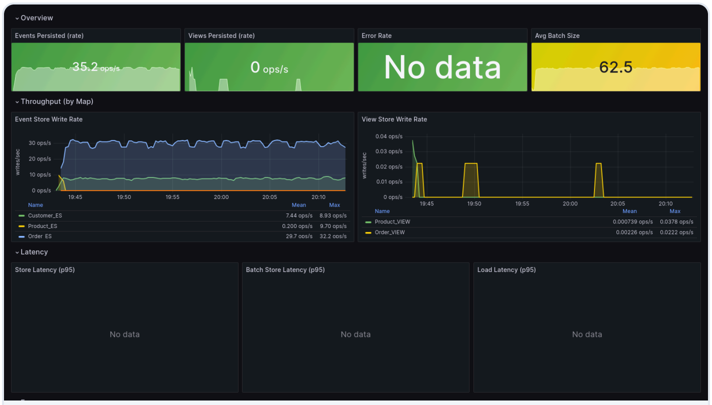
Saga Coordination

54K+ sagas, compensation tracking, step duration percentiles, timeout detection


View Entry Counts
Account View Updates (Total)
13644
Inventory View Updates (Total)
374
Order View Updates (Total)
53670
Payment View Updates (Total)
195
View Update Rates
View Update Rate (by Service)
View Update Rate (by Event Type)
View Staleness & Pipeline Health
View Staleness (End-to-End Latency p95 by Service)
Event Processing Backlog (Received - Processed Rate)

Materialized Views

View rebuild latency, cache hit rates, entry counts, near cache performance



Persistence Layer

Write-behind queue depth, batch sizes, PostgreSQL latency, eviction metrics

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 / ML	Spring AI MCP Server 1.0, VectorCollection
Metrics	Micrometer / Prometheus
Dashboards	Grafana 10.3
Tracing	Jaeger (OTLP)
Containers	Docker / Docker Compose / K8s Helm

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)
mcp-server	AI assistant integration (:8085)

Hazelcast Primitives Used

IMap (event store, views, saga state, outbox, DLQ) • **ITopic** (event bus) • **Jet Pipeline** (stream processing) • **Event Journal** (change stream) • **Entry Processor** (atomic updates) • **FlakeIdGenerator** (sequences) • **Near Cache** (query acceleration) • **MapStore** (write-behind persistence) • **VectorCollection** (similarity search, EE)

Accompanying Blog Series: A 10-part blog series walks through every architectural decision, from event sourcing fundamentals through saga patterns, persistence, AI integration, resilience, and production deployment. Each post maps to a working code milestone in the framework — ideal for customer-facing technical enablement or self-guided learning.

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: Available upon request
Quick Start: ./scripts/docker/start.sh
Demo: ./scripts/demo-scenarios.sh
Contact: mike.yawn@hazelcast.com