System Design

Agentic Orchestration Builder: System Design Document

Build an event-driven, stateful orchestration platform that unifies deterministic DAGs and adaptive multi-agent workflows under a single control plane, with durable execution, human-in-the-loop checkpoints, and full replay/observability. Ground the runtime in durable workflow primitives (e.g., Temporal/Zeebe), overlay agentic coordination patterns for non-determinism, and expose a drag-and-drop builder plus AI copilot for design, governance, and execution.

1) Problem Statement and Vision

Modern automations must handle both:

- Deterministic flows: predictable, repeatable DAGs with clear success/error paths.
- Non-deterministic flows: agentic, context-driven decisions, dynamic branching, reflection/critique loops, and human approvals at critical steps.

The platform must:

- Treat events as the control plane to drive orchestration, communication, and recovery.
- Provide durable state, replay, timeouts, retries, and compensations.
- Natively support agentic patterns (coordinator, hierarchical, parallel/concurrent, iterative refinement, human-in-the-loop) with guardrails and approvals.
- Offer a visual builder and an AI co-pilot that elicits requirements and co-designs robust, secure, cost-aware workflows.

2) Functional Requirements

2.1 Workflow Authoring and Execution

- Visual drag-and-drop builder for triggers, actions, conditions, parallel branches, loops, compensations, and agent steps (deterministic nodes and agentic nodes).
- Import/export BPMN/DMN (optional) for enterprise process teams.
- Event-driven triggers (webhooks, message topics, schedules, file/object events, DB CDC).
- Deterministic execution with durable timers, retries, idempotency, compensations, and saga patterns.
- Non-deterministic/agentic execution with coordinator/manager agents, reflection loops, and human checkpoints.

2.2 Agentic Capabilities

- Multi-agent coordinator to decompose goals and dispatch tasks to specialized agents; supports hierarchical decomposition and concurrent fan-out/fan-in.
- Tool use library (HTTP/API, DB, RPA connectors, scraping, email, chat, files), with scoped credentials per agent (RBAC).
- Context engineering and memory: vector store for embeddings, episodic/execution memory, secure context windows, and per-tenant/flow isolation.
- Safety and governance: policy checks, rate/cost limits, approval gates, risk scoring, and rollback/containment strategies.

2.3 Human-in-the-Loop

- Inline approval, multi-party review, escalation and SLAs, with durable pause/resume and replay.
- “Critical step confirmation” prompts injected by AI copilot at risk-sensitive nodes, configurable by policy.

2.4 Event and State Management

- Event ingestion (webhooks, connectors), enrichment, routing to workflows/agents, dead-letter queues, and ordered/partitioned processing.

- Durable state store for workflow definitions, executions, decisions, timers, and checkpoints; full audit, replay, and time travel.

2.5 AI Copilot (Automation Mode)

- Requirement elicitation: interrogates user for intent, scope, security, data sensitivity, SLAs, costs, margins, ROI.
- Co-architects workflows (deterministic vs agentic), proposes nodes and guardrails, inserts approvals at critical points, and estimates costs/latency.
- Auto-generates test data, simulators, and golden-path/equivocation tests for reliability.

2.6 Integrations and Connectors

- Out-of-the-box connectors: HTTP, Webhooks, DBs, S3/GCS, email/SMS/WhatsApp, calendars, Slack/Teams, CRMs/ERPs, RPA bridges.
- LLM backends: OpenAI/Anthropic/local via adapters for cost/latency control; routing to faster/smaller models for simpler tools.

2.7 Observability and Analytics

- End-to-end tracing with execution graphs, per-node timings, token usage, cost per run, failure clusters, drift detection.
- Policy/reporting dashboards: SLA attainment, approval latencies, cost centres, model/tool health.

2.8 Multi-tenancy and Enterprise Controls

- Hard isolation by tenant with quotas, rate limits, data residency and region pinning, and org-level RBAC/policies.

2.9 Marketplace and Templates

- Reusable node packs, agent roles, starter blueprints for industry use-cases. Certification and signed bundles.

3) Non-Functional Requirements

Category	Target/Requirement
API read latency	P99 < 200 ms

API write latency	P99 < 500 ms
Webhook ingress	< 100 ms processing
Deterministic step latency	< 100 ms median (non-external)
Agentic decision step	0.5–3 s depending on model/tool
Availability	99.9% baseline; HA across AZs; graceful degradation on LLM/provider failures
Durability	11 nines for logs/artifacts via replicated object storage; DB multi-AZ; WAL/backups
Security/Compliance	TLS 1.3, AES-256 at rest, secrets manager, RBAC/ABAC, audit trails, PII detection/masking; SOC2/GDPR-ready
Data residency	Tenant-pinned stores and compute with regionalization
Cost efficiency	Autoscaling, model routing, caching, batch where tolerable; robust cost telemetry

4) Back-of-the-Envelope Estimates

Assumptions (initial public launch):

- 50,000 MAU; 10 workflows/user; 2M executions/day; 10M agent invocations/day; 5M AI API calls/day; peak/avg QPS ≈ 5x.
- Storage growth: ~500 GB/month for execution logs and artifacts (object storage, compressed).
- Peak QPS targets: 5,000 for API+event ingress; DB IOPS target ~15,000 with 85% cache hit.

Generated capacity/cost CSV assets

Interpretation:

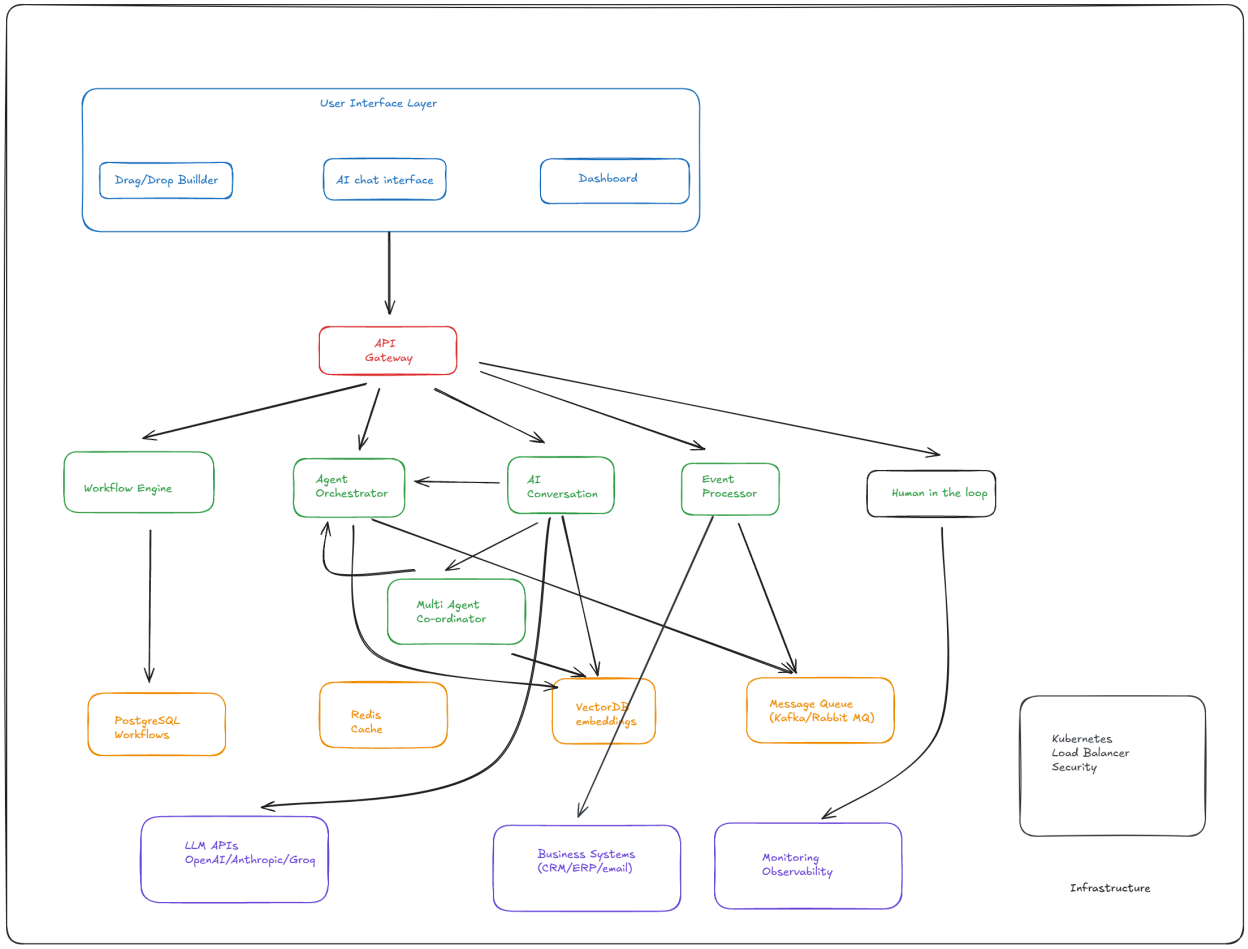
- Kafka partitions sized for 5–10K msg/s with 3x replication; consumer groups per service.
- PostgreSQL with read replicas, connection pooling; consider sharding by tenant_id at >1K tenants per shard benchmark.
- Redis for session/coordination; TTL strategies to keep cache hit ≥85% and reduce DB IOPS.

- Vector DB sized for prompts/memories; per-tenant namespaces; tiered retention.

Rough monthly costs (cloud-managed, blended; varies by region/provider):

- Compute (Kubernetes): $8K; DB: $3.5K; Cache: $0.8K; Storage: $1.2K; Kafka: $0.6K; LLM: $15K; Vector DB: $1.5K; Monitoring: $0.5K; LB/DNS: $0.4K → ~$31.5K/month at this scale.

5) High-Level Architecture



- UI: Drag-and-drop builder; AI chat copilot; Execution dashboards.
- API Gateway: AuthN/Z, rate limiting, request validation, tenant routing.
- Core Services:

    - Workflow Engine (deterministic): durable execution, timers, retries, compensations (Temporal/Zeebe).
    - Event Processing: webhook ingress, enrichment, routing, DLQs, replays; Kafka topics and schemas.
    - Multi-Agent Coordinator: implements coordinator/hierarchical/concurrent patterns; task ledger; cost/latency budgeter; tool adapters.
    - AI Conversation Service: requirement elicitation, planning, critique/reflection, grounding; prompt libraries; secure toolformer calls.
    - Human-in-the-Loop: approval tasks, escalations, SLAs; durable pause/resume; policy injection.
    - Policy & Governance: rules, risk scoring, PEM/OPA evaluation, secrets and vault integration.
    - Connector Hub: outbound/inbound connectors with backpressure, rate controls, token buckets, and provider-specific retry semantics.
- Data Plane:
    - PostgreSQL: tenants, users, workflow_defs, executions, step_executions, approvals, policies, credentials metadata.
    - Redis: sessions, hot keys, distributed locks, short-lived coordination.
    - Vector DB: embedding stores for contexts, memories, tool schemas; per-tenant namespaces.
    - Kafka: event buses per domain (triggers, approvals, agent-tasks, audits, DLQ).
    - Object Storage: artifacts, transcripts, audit bundles, replays.
- Infrastructure:
    - Kubernetes: autoscaling, HPA/VPA, node pools (general, CPU-bound, GPU-bound for local inference).
    - Observability: Prometheus/Grafana, tracing, ELK; SLOs and alerting playbooks.
    - Security: mTLS inter-service, secret management, KMS; egress controls.

Summary:

- Event-driven backbone with Kafka/EventBridge-equivalent for decoupling.
- Deterministic durability via workflow engine (Temporal/Zeebe/iWF).
- Agentic orchestration patterns (coordinator, hierarchical, concurrent, iterative refinement, human-in-the-loop) per cloud/industry references.
- Drag-and-drop builder usability principles.

6) Database Design (core tables)

Tenancy and Identity

- tenants(id, region, plan, data_residency)
- users(id, tenant_id, email, role, mfa, status)
- service_accounts(id, tenant_id, scopes)
- api_keys(id, tenant_id, hash, rate_limit, expires_at)

Workflow Model

- workflow_definitions(id, tenant_id, name, version, spec_json, type: deterministic|agentic|hybrid, active, created_at, created_by)
- workflow_executions(id, workflow_id, tenant_id, status, current_step, started_at, completed_at, context_json, cost_cents)
- step_executions(id, execution_id, step_index, step_type, status, input_json, output_json, error, started_at, completed_at)
- schedules(id, tenant_id, cron, workflow_id, enabled)
- webhooks(id, tenant_id, source, secret, filters_json, enabled)

Agentic Model

- agents(id, tenant_id, role, tools_json, policy_json, status, last_active_at)
- agent_tasks(id, tenant_id, agent_id, kind, priority, payload_json, status, scheduled_at, started_at, completed_at, result_json)
- task_ledger(id, execution_id, parent_id, title, state, assigned_agent_id, cost_budget_cents, token_budget, reason_json)

Approvals and Policy

- approval_tasks(id, tenant_id, execution_id, step_id, approvers_json, sla_seconds, status, decided_by, decided_at, decision, comment)
- policies(id, tenant_id, name, type, rules_json, enabled)
- secrets_metadata(id, tenant_id, name, scope, rotated_at)

Events and Audit

- events(id, tenant_id, type, source, payload_json, correlation_id, status, created_at, processed_at)
- audits(id, tenant_id, actor_type, actor_id, action, resource_type, resource_id, diff_json, ts)

Analytics/Quotas

- usage_counters(id, tenant_id, metric, window_start, window_end, count, cost_cents)
- sla_breaches(id, tenant_id, execution_id, kind, details_json, ts)

Vector Store

- Stored in external vector DB with metadata: tenant_id, workflow_id, execution_id, type (memory, tool_doc, KB), embedding, pointer_uri.

Indexes and partitioning:

- Composite indexes for tenant_id + status/stage columns; time-partition large append-only tables (events, audits, step_executions) for pruning.
- Consider sharding by tenant_id for very large tenancy cohorts.

7) API Design (some examples)

Auth

- POST /v1/auth/token
- GET /v1/me

Workflows

- GET /v1/workflows?tenant=...&active=true

- POST /v1/workflows {name, type, spec}
- POST /v1/workflows/{id}/execute {input, correlation_id}
- GET /v1/executions/{id}
- POST /v1/webhooks/ingest (HMAC-verified)

Agentic

- POST /v1/agents {role, tools, policy}
- POST /v1/agents/{id}/tasks {payload}
- GET /v1/executions/{id}/ledger
- POST /v1/executions/{id}/reflect (trigger critique/replan)

Human-in-the-Loop

- GET /v1/approvals?status=pending
- POST /v1/approvals/{id}/decision {approve|reject, comment}

Governance

- POST /v1/policies {rules}
- GET /v1/costs?window=...

Observability

- GET /v1/executions/{id}/graph
- GET /v1/metrics/executions?group_by=workflow
- GET /v1/traces/{trace_id}

Webhook security: HMAC signature, replay windows, IP allowlists.

Pagination: cursor-based; idempotency keys for create/execute.

Rate limits: per-tenant tier; burst allowances; per-key quotas.

8) Orchestration Patterns Supported

- Deterministic: sequential, parallel fan-out/fan-in, loop, compensation/saga, timeouts; best for predictable tasks.

- Coordinator pattern: AI manager routes tasks to specialized agents; adaptive branching; tool-based actions.
- Hierarchical task decomposition: multi-level planning for ambiguous problems, with higher cost/latency trade-offs.
- Concurrent orchestration: multiple agents in parallel with aggregation and selection.
- Iterative refinement and review/critique loops: quality gates and self-correction.
- Human-in-the-loop: approval and escalation pattern; policy-driven checklists.

9) Execution Semantics and Safety

Deterministic steps:

- Pure functions with retries and backoff; idempotent side effects; compensations for non-idempotent actions; exactly-once intent with outbox pattern.

Agentic steps:

- Budgeted token/cost ceilings; timeouts; tool call whitelists; sandboxed outputs; pre/post validation hooks; red-teaming prompts; auto-insert approval on high-risk scopes.
- Fall-back strategies: smaller/cheaper models, cached responses, human escalation.

Event handling:

- Ordering where needed via partition keys; DLQs with reprocess UI; poison-pill detection; circuit breakers.

10) Observability and SLOs

Metrics

- Request rate/latency/error per API; execution duration per workflow; step distribution; queue lag; cache hit; DB latency; token/cost per execution; model/provider latency.

Tracing

- Distributed tracing with correlation IDs across workflow, agent, tool calls.

Logging

- Structured logs with redaction for PII; long-term archival tiers with lifecycle policies.

SLOs (starter)

- API availability 99.9%; execution success rate ≥ 99% for deterministic; mean approval latency SLA per policy; webhook processing P95 < 100 ms.

Alerts

- Error rate spikes; queue lag; approval SLA breaches; cost budget overruns; model timeouts and provider saturation.

11) Security, Compliance, and Governance

- RBAC/ABAC for users, service accounts, and agents; resource-scoped permissions and allowlisted tools.
- Data protection: encryption in transit and at rest, field-level encryption for sensitive fields, PII detection and masking.
- Tenant isolation: network and data segregation; per-tenant encryption contexts; region pinning; key rotation.
- Governance and audit: immutable audit trails for user/agent actions, prompt and tool logs, replay artifacts; policy packs per vertical.

12) Deployment and Scaling

- Kubernetes with autoscaling; distinct pools for CPU vs GPU; canary/blue-green with metrics gates.
- Horizontal scaling by microservice; Kafka partitions sized to traffic domains; Redis clustering; DB read replicas; shard by tenant at scale.
- Multi-region: active-active stateless, active-passive for DBs; cross-region replication with bounded staleness; tenant-level residency controls.

13) Migration and Reliability Practices

- Zero-downtime schema migrations (expand/migrate/contract); dual writes and backfills with validation.
- Chaos testing in staging for workflow timeouts, DLQs, and agentic fallbacks.
- Replay tooling from archived events with deterministic vs agentic semantic flags.

14) Cost Optimization

- Model routing/caching; response reuse for deterministic prompts; toolformer-style planner to minimize LLM calls; summarize/condense contexts.
- Selective tracing/logging levels; lifecycle policies for storage; spot/interruptible nodes for batch.
- Connector rate shaping and batching to cut provider costs.
- Per-tenant cost dashboards and budgets with automatic throttling/escalation.

15) Product UX Highlights

- Builder: minimal-text, visual-first canvas with node packs (deterministic, agent, human, event, compensations), inline cost/latency estimates per path, and policy hints.
- Copilot: requirements interview, drafts initial workflow, highlights critical steps needing confirmation, simulates runs with synthetic event streams (no PII), recommends guardrails and tests.
- Execution view: live swimlanes (agents, human, systems), replay slider, and “what-if” branching.
- Templates: industry blueprints for sales ops, support triage, finance approvals, supply chain exceptions.