AZURE AI ARCHITECTURE · REFERENCE BLUEPRINT

Sphera AI Platform PoC → Production Blueprint

Full system blueprint to move the Sphera AI PoC to production on Microsoft Azure — component mapping, target architecture, code-issue remediation playbook, decision matrices, security & governance, and a 12-week migration roadmap.

Azure · Claude / Anthropic · RAG · Databricks · AI-DLC · Hermes Agents

---

01 Executive Brief — Situation

Why the PoC is stuck, what "production-ready" means for AI on Azure, and the outcomes this blueprint delivers.

PoC	Production	12 wks	#1 block
Current state	Target state	Migration window	Code quality & sec

The Situation

as reported

- **Company:** Sphera — enterprise ESG, sustainability & operational-risk software (data-heavy, compliance-sensitive).
- **Platform:** Microsoft Azure is the approved cloud. AI PoC is built but **cannot move to production**.
- **Current components:** Claude (Anthropic), nginx, AI-DLC (AI-Driven Development Life Cycle), Hermes agents, Git, RAG, Pendo, Juno*, Jina (embeddings/rerankers), Databricks, MS Teams.
- **Symptom:** "Currently having code issue" — PoC code does not meet production bar (see Section 04 triage).
- *Juno assumed to be an internal AI assistant / orchestration tool — confirm vendor & role in the Assumptions Register (§02).

Why PoCs stall (pattern): notebook/monolith code, hardcoded keys, no retry/rate-limit handling, no auth, no evals, no cost guardrails, direct vendor API calls, no IaC. The fix is architectural, not a code patch — this handbook is that fix.

What Production-Ready Means Here

Operable

Deploys repeatably (IaC), runs on containers/serverless, zero hardcoded secrets, health checks, autoscaling.

Observable

Every prompt, token, cost, latency, and eval trace is logged; alerts fire on drift & error rate.

Governed

Entra ID + RBAC, private networking, data residency, AIDLC stage gates, Responsible-AI guardrails.

Tested

Unit + integration + golden-set evals in CI; prompt & model versions pinned; rollback in minutes.

Cost-Controlled

Budget alerts, token caps, caching, model tiering (haiku/sonnet/opus by task).

User-Loved

Pendo in-app guidance + feedback loops; Teams human-in-the-loop approvals for high-stakes actions.

02 Current-State Component Map

As-Is

Every named component, its role today, and how it maps into the production blueprint. Colors = target layer.


Users & Channels
Web Portal
MS Teams (AI/Copilot)
Pendo (guidance)
Edge / Gateway
nginx (reverse proxy)
⚠ no WAF · no rate limits · no mTLS
AI / Agent Layer
Hermes agents
Juno* (assumed)
RAG pipeline (Jina emb/rerank)
⚠ no guardrails · no tracing
Data & Governance
Databricks Lakehouse
Git (code)
AI-DLC lifecycle (gates)
⚠ secrets in repo · no IaC
Channels
Gateway
AI/ Agent
Data/ Gov

Component → Production Mapping

12 components

Component	Role in PoC	Production Target on Azure	Action
Claude / Anthropic	LLM inference via direct API	Claude via **Azure AI Foundry** (enterprise control) — direct API as fallback	**Migrate**
nginx	Reverse proxy	Azure Front Door + WAF at edge; **Azure API Management (APIM)** as gateway; nginx kept only for internal sidecars	**Replace/ split**
AI-DLC	AI Development Life Cycle process	Formalize as stage-gate pipeline in Azure DevOps (define→data→build→verify→deploy→monitor)	**Formalize**
Hermes agents	Agent orchestration & automation	Containerized agent runtime on Azure Container Apps with Managed Identity + MCP tool registry	**Containerize**
Git	Source control	Azure DevOps / GitHub Enterprise — branch policy, PR gates, semantic releases	**Harden**
RAG	Retrieval-augmented generation	Azure AI Search (hybrid + semantic) with Jina embeddings/reranker; chunking & eval pipeline	**Rebuild**
Pendo	Product analytics & in-app guidance	Keep — feed AI feature telemetry, feedback, and NPS into eval & improvement loop	**Keep**
Juno*	Assumed internal assistant/ orchestrator	Confirm vendor; fold into agent layer as an internal copilot surface	**Verify**
Jina (jini)	Embeddings / rerankers	Keep for retrieval quality (or swap to Azure OpenAI text-embedding-3) — decision matrix §06	**Evaluate**
Databricks	Lakehouse, pipelines, ML	Keep as data source of truth; Unity Catalog for lineage/ACL; sync to Azure AI Search	**Integrate**

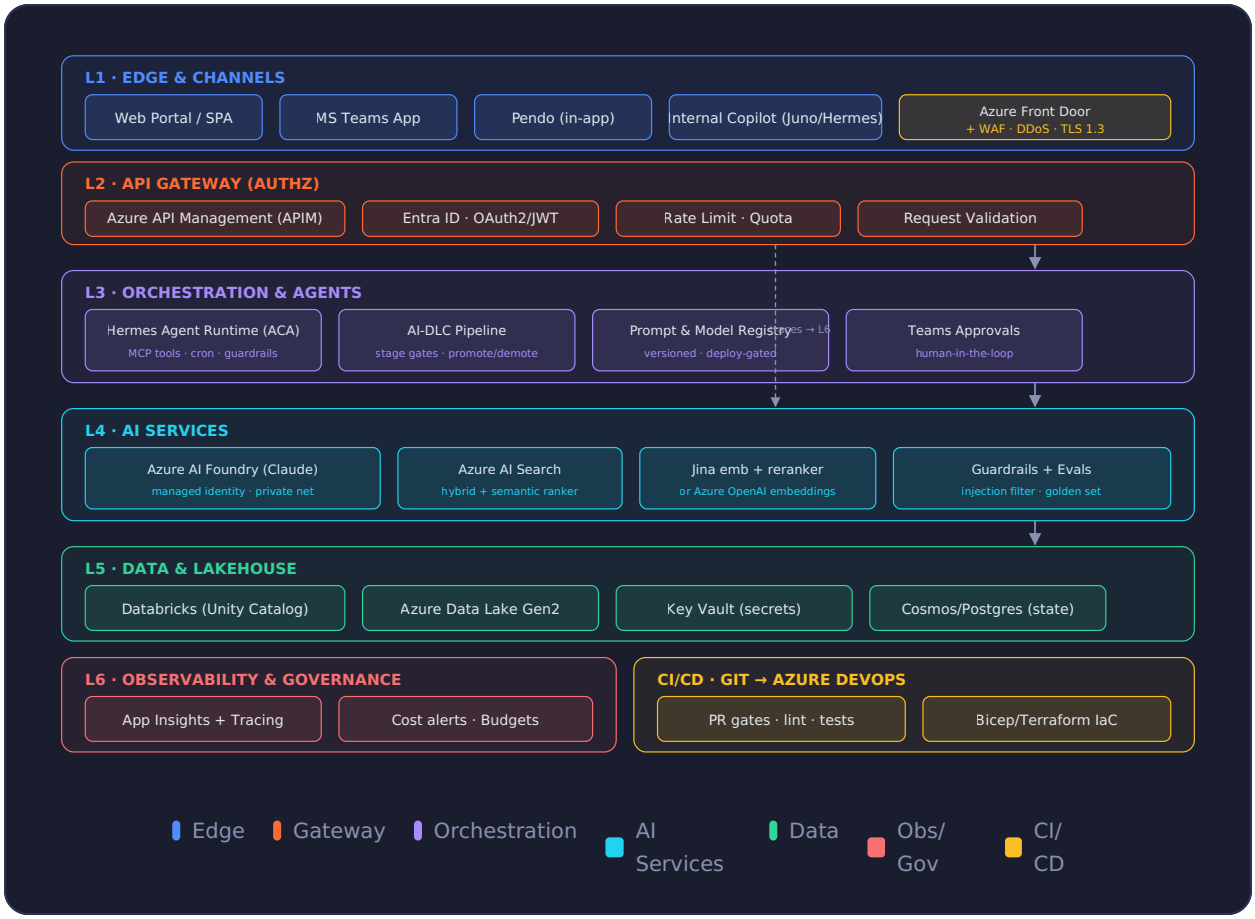
MS Teams (msteai)	Collaboration	Teams app with Adaptive Cards for human-in-the-loop approvals + bot notifications	**Extend**
Azure	Cloud platform	Everything above lands on Azure: Entra ID, Key Vault, ACA/AKS, Monitor, AI Foundry	**Core**

Assumptions Register: (1) **Juno** = internal AI assistant/orchestration tool — confirm vendor & API surface. (2) **Jina** = Jina AI embeddings-v3 / reranker-v1 used in the RAG stack. (3) **AI-DLC** = AI-Driven Development Life Cycle governance framework. (4) Sphera data is compliance-sensitive (ESG/emissions/risk) → data residency & PII controls are first-class requirements. (5) "msteai" = MS Teams + AI/Copilot surface.

03 Target-State Azure Blueprint

To-Be

The full production architecture. Six layers: Edge → Gateway → Orchestration → AI Services → Data → Observability/Governance, with CI/CD running alongside.



Layer-by-Layer Guidance

the blueprint

Layer	Key Azure services	Why	Replaces / upgrades
L1 Edge	Front Door + WAF, CDN	Global TLS, DDoS, bot protection; single public entry	Bare nginx exposure
L2 Gateway	APIM, Entra ID	OAuth2/JWT, rate limits, quotas, policy-based routing, per-key usage	nginx as API gateway
L3 Orchestration	Azure Container Apps, Dapr, Service Bus	Agent runtime w/ managed identity, event-driven scale-to-zero, durable workflows	Ad-hoc scripts & notebooks
L4 AI Services	Azure AI Foundry (Claude), AI Search, Jina/embeddings, Azure OpenAI	Enterprise LLM control plane: private networking, entitlements, content filters, evals	Direct Anthropic API calls
L5 Data	Databricks + Unity Catalog, Data Lake, Key Vault, Cosmos/ Postgres	Single source of truth; lineage; secrets never in code	Scattered CSVs / hardcoded keys
L6 Observability	App Insights, Log Analytics, Grafana (optional), Pendo, cost alerts	Prompt/token/cost/ latency traces, eval dashboards, drift alarms	print() debugging
CI/CD	Azure DevOps / GitHub Actions, Bicep	Everything as code: infra, prompts, models, data pipelines	Manual deploys

Key architectural decision: Put **Claude behind Azure AI Foundry** (same models, enterprise controls — private endpoints, managed identity, audit, entitlements, content safety). Keep the direct Anthropic API as a **failover route** behind the model-abstraction layer. This is the single highest-leverage move for a compliance-sensitive company.

04 Code Triage Playbook — The "Code Issue"

Ten blockers that keep AI PoC code out of production, each with a symptom → root cause → fix pattern → guardrail. This is the section to hand the engineering team.

Blocker #1 — Hardcoded Secrets & API Keys

P0 · Security

- **Symptom:** API keys in `.py` / `.env` committed to Git; keys rotate and break the PoC.
- **Root cause:** Fast PoC iteration, no secret hygiene.
- **Fix:** Azure Key Vault + Managed Identity (`DefaultAzureCredential`) — **zero keys in code**.

```
#    PoC: key in code / .env
ANTHROPIC_API_KEY = "sk-ant-..."

#    Prod: identity-based, no secret in code
from azure.identity import DefaultAzureCredential
from azure.keyvault.secrets import SecretClient

cred = DefaultAzureCredential()   # Managed Identity in prod, dev login locally
client = SecretClient(vault_url="https://sphera-ai.vault.azure.net/", credential=cred)
ANTHROPIC_API_KEY = client.get_secret("anthropic-api-key").value
```

> **Guardrail:** secrets scanning in CI (gitleaks/trufflehog) + `git filter-repo` to purge history + deny-list file patterns in branch policy.

Blocker #2 — No Retry / Rate-Limit Handling

P0 · Reliability

- **Symptom:** Random 429/5xx from Anthropic under load; jobs die mid-run; users see errors.
- **Root cause:** Raw `client.messages.create()` with no retry, no backoff, no fallback model.
- **Fix:** Retry with exponential backoff + jitter, circuit breaker, and a model/route fallback chain.

```
from tenacity import retry, stop_after_attempt, wait_exponential_jitter,
retry_if_exception_type
import httpx

@retry(
    stop=stop_after_attempt(5),
    wait=wait_exponential_jitter(initial=1, max=60),
    retry=retry_if_exception_type((httpx.HTTPStatusError, ConnectionError)),
    reraise=True,
)
def call_llm(model: str, messages: list[dict]) -> str:
    # ... invoke via AI Foundry gateway, fallback to Anthropic direct on timeout
    return response

# Circuit breaker: if error_rate > 10% in 60s → route to fallback model for 5 min
```

> **Guardrail:** never call the LLM provider directly from app code — always through a gateway/client wrapper that owns retry, fallback, budget, and tracing.

Blocker #3 — Monolith Notebook / Script Sprawl

P1 · Maintainability

- **Symptom:** One giant notebook or `main.py` ; nobody can test it; imports are a tangle.
- **Root cause:** PoC written as a linear experiment, never refactored.
- **Fix:** Package by layer: `ingest/ · retrieval/ · llm/ · agents/ · api/ · eval/` , typed interfaces, no business logic in notebooks.

```
sphera-ai/
├── pyproject.toml           # pinned deps (uv/poetry)
├── src/sphera_ai/
│   ├── config.py            # pydantic-settings (env-driven, validated)
│   ├── retrieval/           # chunking, embeddings, search
│   ├── llm/                 # gateway client, prompt templates, fallback
│   ├── agents/              # Hermes/tool orchestration, MCP registry
│   ├── api/                 # FastAPI routes (thin)
│   └── eval/                # golden sets, scorers
└── tests/                   # unit + integration + eval (run in CI)
```

> **Guardrail:** 80% coverage gate in CI; notebooks only as analysis artifacts, never as the runtime.

Blocker #4 — No Input/Output Validation

P1 · Robustness

- **Symptom:** LLM returns malformed JSON → `KeyError` crashes; garbage in → hallucinated garbage out.
- **Root cause:** Free-form `json.loads()` on model output.
- **Fix:** Structured output (tool-use / Pydantic) + retry-on-parse-failure.

```
from pydantic import BaseModel, Field
from typing import Literal

class TradeSignal(BaseModel):
    action: Literal["buy", "sell", "hold"]
    confidence: float = Field(ge=0.0, le=1.0)
    reasoning: str

# model called with response_format / tool schema → parse with .model_validate()
# on ValidationError → one retry with "fix your output" prompt, else degrade to hold
```

Blocker #5 — Prompt Injection & Untrusted Inputs

P0 · Security

- **Symptom:** RAG documents or user fields containing "ignore previous instructions"; model exfiltrates data.
- **Root cause:** No isolation between instructions, data, and user input; no output filtering.
- **Fix:** Delimiters + instruction hierarchy, input sanitization, output guardrails (PII/secret regex), least-privilege tool access, human approval for high-impact tool calls.

> **Guardrail:** treat **all** retrieved content as untrusted data, never instructions. Tools must re-validate arguments server-side. Log every tool call.

Blocker #6 — No Observability / Tracing

P1 · Ops

- **Symptom:** "It worked yesterday" — no logs, no traces, no token/cost accounting.
- **Root cause:** `print()` debugging in a serverless world.
- **Fix:** Structured logging + OpenTelemetry spans per request: prompt hash, model, tokens, latency, cost, retrieval ids.

```
import logging, time, opentelemetry.trace as trace

tracer = trace.get_tracer("sphera-ai")
logger = logging.getLogger("sphera_ai.llm")

with tracer.start_as_current_span("llm.call") as span:
    t0 = time.perf_counter()
    resp = call_llm(model, messages)
    span.set_attributes({
        "llm.model": model, "llm.prompt_hash": sha256(str(messages)),
        "llm.tokens_in": resp.usage.input_tokens, "llm.tokens_out":
resp.usage.output_tokens,
        "llm.latency_ms": int((time.perf_counter()-t0)*1000),
        "llm.cost_usd": estimate_cost(model, resp.usage),
    })
logger.info("llm_call", extra={"model": model, "tokens": resp.usage})
```

> **Guardrail:** every LLM call is traceable to (user, request, prompt version, model, cost) — this is also your audit story for compliance.

Blocker #7 — Unversioned Prompts & Models

P1 · Reproducibility

- **Symptom:** Results change between runs; can't roll back a bad prompt; "which model was this?"
- **Root cause:** Prompts inline in code, model names hardcoded, no registry.
- **Fix:** Prompt templates as versioned artifacts in Git (or a prompt registry); model versions pinned in config; prompt/model deploy = release with rollback.

```
# config: pinned, environment-specific
model:
  primary: "claude-sonnet-4-6@2026-06-01"   # pinned via AI Foundry deployment
  fallback: "claude-haiku-4-5@2026-06-01"
  embedding: "jina-embeddings-v3@latest-validated"

prompt_version: "summary-v2-2026-08-01"
```

Blocker #8 — No Tests / No Evals

P1 · Quality

- **Symptom:** Every change is a gamble; regressions discovered by users.
- **Root cause:** "Evals are hard" → skipped.
- **Fix:** Golden-set evals in CI (50–200 curated Q/A with rubric scorers), LLM-as-judge with guardrail, regression gates on answer quality, latency, cost, hallucination rate. Start small; grow the set with Pendo-flagged bad answers.

> **Pattern:** eval results are the acceptance test for promoting a prompt/model. No eval improvement → no deploy (AIDLC gate).

Blocker #9 — Unbounded Cost & Latency

P2 · Cost

- **Symptom:** Bill spikes; slow responses; users complain.
- **Root cause:** Huge contexts, no caching, no model tiering, no budgets.
- **Fix:** Prompt caching (Anthropic + AI Foundry support), model tiering by task (haiku for classification, sonnet for generation), token caps, response caching for identical queries, Azure budget alerts at 50/80/100%.

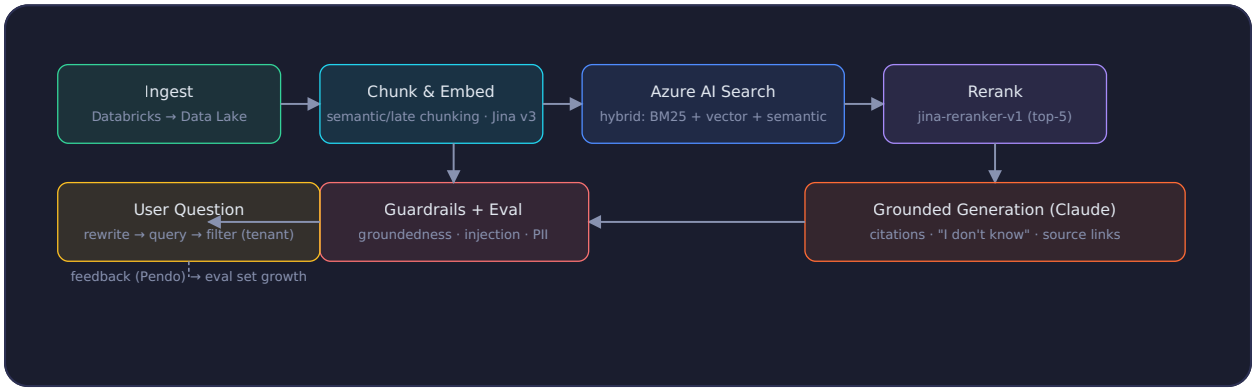
Blocker #10 — No Auth / No RBAC P0 · Security

- **Symptom:** Anybody with the URL can query the API; internal data exposed.
- **Root cause:** PoC had no identity layer.
- **Fix:** Entra ID SSO (OAuth2 PKCE for SPA), APIM validates JWT, role-based authorization per capability (viewer/analyst/admin), per-tenant data isolation checks.

Triage order: P0s first (#1, #2, #5, #10 — security & reliability), then P1s (#3, #4, #6, #7, #8), then P2 (#9). Each fix lands as a PR with a test and a guardrail — that is how the PoC earns "production" in 4–6 weeks.

05 RAG Pipeline Design Retrieval

Production RAG: ingestion → chunking → embeddings (Jina) → hybrid search (Azure AI Search) → rerank → grounded generation with citations and evals.



RAG Design Decisions gotchas

Decision	Recommendation	Rationale
Chunking	Semantic chunking (150–400 tokens), metadata-rich; late chunking for long docs	Better retrieval than fixed-size; keeps citations intact
Search	Hybrid (BM25 + vector + semantic ranker) via Azure AI Search	ESG docs mix terminology & exact IDs — lexical+semantic beats either alone
Embeddings	Jina embeddings-v3 (multilingual, Matryoshka) or Azure OpenAI text-embedding-3-large	Multilingual ESG content; evaluate on your golden set (§06 matrix)
Rerank	jina-reranker-v1 on top-20 → top-5	+10–20% answer quality for a few ms; cheap insurance
Tenancy	Index per tenant + filterable `tenant_id` field; RBAC enforced in app	Compliance — Sphera data is per-customer sensitive
Freshness	Incremental indexer from Databricks (Change Data Capture); daily + on-demand	Emissions/safety data must not go stale
Evals	Golden set 50→200 Q/A; metrics: faithfulness, answer relevance, context precision/recall, citation accuracy	Every RAG change is a PR with an eval delta

Azure AI Search vs Databricks vector search: AI Search wins for the RAG serving path (hybrid + semantic + SLA + managed). Databricks stays the compute/curation plane. Keep **one** canonical index writer owned by the ingestion pipeline.

06 Decision Matrices Trade-offs

Seven build-vs-buy / this-vs-that decisions the team is facing right now. Win = recommended choice for Sphera's context.

M1 · Model Access: Azure AI Foundry vs Direct Anthropic API

Factor	AI Foundry (win)	Direct Anthropic API
Enterprise controls	**Private endpoints, entitlements, audit**	**Vendor-only controls**
Compliance (ESG data)	**Azure compliance + data residency story**	**Separate DPA/ compliance scope**
Managed identity / networking	**First-class Azure integration**	**Keys + egress needed**
Feature velocity	**Slight lag behind Anthropic releases**	**Same-day new models**
Cost	**Azure commitments/EA discounts**	**List price**
Verdict	**Primary = AI Foundry; keep direct API as failover behind abstraction layer**	

M2 · Gateway: Azure APIM vs Keep nginx

Factor	APIM (win)	nginx only
OAuth2/JWT validation	**Native Entra integration**	**Manual/openresty work**
Rate limits & quotas	**Per-key, per-plan, built-in**	**Custom Lua/limits**
Analytics per consumer	**Built-in + export to App Insights**	**Custom access-log parsing**
Operational simplicity	**Another managed service**	**Already running**
Verdict	**APIM for external/internal API surface; nginx stays only as an ingress sidecar (e.g., local dashboards)**	

M3 · Compute: Azure Container Apps vs AKS vs Azure Functions

Factor	Container Apps (win)	AKS	Functions
Team size / K8s expertise	**No cluster management**	**Full cluster ops**	**Zero infra**
Long-running agents / workflows	**Durable, scale-to-zero, Dapr**	**Full control**	**Execution limits**
Managed identity + VNet	**Built-in**	**Via workload identity**	**Built-in**
Cost at low volume	**Scale-to-zero**	**Node minimums**	**Per-execution**
Verdict	**ACA for the agent/API tier; Functions for event-driven ingestion triggers; AKS only if the team grows a platform team**		

M4 · Vector Search: Azure AI Search vs Databricks Vector Search

Factor	AI Search (win for serving)	Databricks Vector Search
Hybrid + semantic ranking	**BM25 + vector + semantic out of the box**	**Vector only (v recent)**
Serving SLA / latency	**Managed search SLA, low latency**	**Compute-dependent**
Ecosystem (indexers, skills)	**Azure-native indexers, enrichments**	**Delta Lake integration great**
Verdict	**AI Search = serving index; Databricks = curation & feature plane feeding it**	

M5–M7 · Quick Calls

Decision	Recommendation	Why
M5 · Embeddings: Jina vs Azure OpenAI	**Evaluate both on your golden set; Jina if multilingual/ late-chunking matters**	Quality delta is domain-specific — let evals decide, keep provider behind an interface
M6 · Secrets: Key Vault vs .env	**Key Vault + Managed Identity**	Non-negotiable for compliance (also feeds §04 blocker #1)
M7 · Observability: App Insights + Langfuse-style tracing vs DIY	**Managed tracing (Langfuse Cloud/OSS or Azure Monitor GenAI tracing)**	Prompt/token/cost trace out of the box; DIY logs drift and die

07 Security & Governance

Guardrails

Identity, network, data, and process controls — the layer that unlocks production approval for compliance-sensitive AI.

Identity & Access

Entra ID SSO (PKCE) for apps; APIM JWT validation; RBAC roles: AI-Viewer / AI-Analyst / AI-Admin; Managed Identity for all service-to-service calls; Conditional Access for admin.

Network

Hub-spoke VNet; Private Endpoints for AI Foundry, AI Search, Key Vault, Databricks; Front Door + WAF (OWASP rules, geo-filter); no public egress except allowlisted providers.

Data Protection

Data classification (public/internal/confidential/PII); PII redaction before LLM context; per-tenant isolation; retention & deletion policies; Azure Purview for lineage.

AI-Specific Controls

Prompt-injection filters; output guardrails (PII, secrets, prohibited content); tool-call allowlists; human approval for write actions; audit log of every model call & tool invocation.

AI-DLC Stage Gates (production approval = gates passed)

Gate	Exit criteria	Owner
G1 · Define	Use case, success metrics, risk tier, data inventory signed off	Product + Architect
G2 · Data	Data sources catalogued; PII flagged; lineage in Unity Catalog; consent/residency verified	Data Eng
G3 · Build	Code passes lint/tests/secrets scan; prompts versioned; abstraction layer in place	Engineering
G4 · Verify	Golden-set evals green; security review; load test; cost estimate approved	QA + Security
G5 · Deploy	Blue/green or canary; rollback runbook; runbook tested in dry-run	Platform
G6 · Monitor	Dashboards live; drift & cost alerts; feedback loop (Pendo) feeding eval set; monthly review	Ops + Product

> **Pattern:** a model/prompt "promotion" is a release artifact that must pass G4 before touching production traffic. This is how "PoC → Prod" stops being an opinion and becomes a checklist.

Compliance Checklist (ESG/enterprise context)

- ☐ Data residency — region pin (e.g., EU/US) enforced at storage & inference
- ☐ DPA with Anthropic (and Azure) covering sub-processors
- ☐ Zero-retention or short-retention model settings where applicable
- ☐ Audit log: who asked what, which model answered, what was retrieved
- ☐ Model cards & system cards for each deployed use case (Responsible AI)
- ☐ Incident response runbook for prompt-injection / data-exposure events
- ☐ Pen test + threat model before GA; quarterly red team of the RAG path

08 Observability & Operations

Run It

What to measure, what to alert on, and how Pendo + Teams close the loop between users, evals, and model improvement.

The AI Service Dashboard (one pane)

< 3s	> 99%	< 2%
P95 latency	Groundedness	Error rate

$ / day	NPS / CSAT
Cost per use case	From Pendo

- **Latency:** per-model, per-feature; alert on P95 breach.
- **Quality:** online evals (LLM-as-judge on sampled traffic) + offline golden set each deploy.
- **Cost:** per use case, per tenant; budgets with 50/80/100% alerts; token caps per key.
- **Drift:** input distribution, retrieval hit-rate, "I don't know" rate, hallucination proxy (groundedness score).
- **Feedback:** Pendo thumbs up/down + comments → weekly triage → eval set growth → model/prompt improvement (AIDLC loop).

Teams as the Control Surface

- Adaptive Cards for **human-in-the-loop approvals** (e.g., agent-proposed actions, sensitive queries).
- Bot notifications: eval regressions, cost spikes, drift alarms, weekly AI report.
- Prompt-injection or PII incident → Teams channel with severity + runbook link.
- Users ask in Teams → same governed pipeline (APIM → agents → AI Foundry). One platform, one audit trail.

Incident & Rollback Runbook (summary)

1. **Detect:** error-rate / cost / groundedness alert fires (App Insights → Teams).
2. **Triage:** trace lookup by request id; classify (model, prompt, retrieval, infra).
3. **Mitigate:** feature flag off / rollback to previous prompt+model release (registry = instant).
4. **Post-mortem:** add regression to golden set; AIDLC gate re-run before re-promotion.

09 Migration Roadmap — 12 Weeks

Four phases, each with exit criteria. The PoC is production-live by week 12 with zero re-architecture — only hardening.

Phase 1 · Foundation (W1–2)

IaC landing zone (Bicep): VNet, Key Vault, Entra app regs, APIM, AI Foundry workspace, budgets. Secrets scan + git history purge. **Exit:** empty environment deploys from one command.

Phase 2 · Harden Code (W3–6)

Refactor into `sphera-ai` package; P0 fixes (#1,#2,#5,#10); abstraction layer; tests + golden-set evals in CI; prompt registry. **Exit:** CI green on every PR; eval baseline recorded.

Phase 3 · Migrate (W7–10)

Move LLM calls to AI Foundry (fallback chain); rebuild RAG on AI Search + Jina; Databricks → incremental indexer; teams approvals; tracing live. **Exit:** parallel-run parity (95% answer match, no PII leaks).

Phase 4 · GA & Scale (W11–12)

Canary → 100% traffic; SLOs + dashboards + runbooks; Pendo feedback loop live; security review + pen test sign-off; cost monitoring in place. **Exit:** production approval (G1–G6 all green).

Parallel-run trick: run PoC and production side-by-side on the same traffic, compare answers with an LLM-judge. "95% parity" is the evidence that convinces stakeholders — much stronger than a slide.

10 Glossary & Cheat Sheet

Reference

Key terms in 60 seconds, then the one-page cheat sheet for daily reference.

Azure AI Foundry

Azure's unified AI platform — model catalog (incl. Claude), deployments, entitlements, evals, content safety.

AI-DLC

AI-Driven Development Life Cycle — stage-gate framework (Define→Data→Build→Verify→Deploy→Monitor) for governing AI releases.

APIM

Azure API Management — gateway with auth, rate limits, quotas, policies, analytics.

RAG

Retrieval-Augmented Generation — ground model answers in your data via retrieval, reducing hallucination.

Managed Identity

Azure service identity with automatic credential rotation — replaces API keys in code.

Golden Set

Curated Q/A test set with rubric scoring — the acceptance test for prompt/model changes.

Groundedness

Eval metric: is the answer supported by retrieved context? (hallucination inverse)

MCP

Model Context Protocol — standard for connecting agents to tools/data (Hermes uses MCP tool registry).

Hybrid Search

Keyword (BM25) + vector search combined — best for mixed terminology/data (ESG docs).

Human-in-the-Loop

Approval step for high-stakes agent actions — e.g., Teams Adaptive Card before a write operation.

Unity Catalog

Databricks governance — lineage, ACLs, and data discovery across the lakehouse.

Scale-to-Zero

Serverless containers that shut down when idle — cuts PoC-stage cost to near zero.

One-Page Cheat Sheet

The printable quick reference (also standalone at /sphera-cheatsheet — download PDF, or use the print button):

Rule	One-liner
#1 Secrets	Never in code → Key Vault + Managed Identity
#2 Calls	Always via gateway wrapper: retry · fallback · budget · trace
#3 Models	Pin versions; registry; deploy = release with rollback
#4 Prompts	Versioned artifacts; treat retrieved data as untrusted
#5 Output	Pydantic/structured; validate; retry-on-parse; degrade safely
#6 Eval	Golden set in CI; no eval improvement → no deploy
#7 Trace	Prompt hash, tokens, cost, latency per call → App Insights
#8 Tenancy	Per-tenant isolation at index + RBAC + query filter
#9 Cost	Model tiering + caching + budgets at 50/80/100%
#10 Gate	Nothing to prod without AIDLC G1–G6 sign-off

Bottom line: the PoC is stuck on code quality, not on the idea. Fix the ten blockers (4 weeks), stand up the target architecture with IaC (2 weeks), migrate behind a fallback chain and prove parity (4 weeks), then GA with evals, budgets, and runbooks (2 weeks). Twelve weeks from "cannot move to prod" to a governed, observable, cost-controlled AI platform on Azure.
