# Agentic AI Control Plane on Azure: A Capability-Driven Architecture for Assist, Orchestrate, Retrieve, and Govern

Dr. Freeman A. Jackson

2025

**Abstract**

This whitepaper presents an integrated reference architecture for enterprise-grade, agentic AI systems built on Microsoft Power Virtual Agents (Copilot Studio), the Microsoft Agent Framework, Semantic Kernel, LangGraph, and Azure-native services. At the core of this architecture is an AI Capability Map organized around four pillars: **Assist** (user experience and channel integration), **Orchestrate** (planning, reasoning, and tool execution), **Retrieve** (knowledge and data access via Retrieval-Augmented Generation), and **Govern** (security, compliance, cost, and lifecycle assurance). A set of supporting views—stakeholder & role maps, value stream, C4 system diagrams, trust boundaries, BPMN flows, state machines, data flows, and GRC overlays—collectively describe how to design, implement, and operate a secure, observable, and cost-controlled AI control plane.

## Contents

## 1. Introduction

Enterprises are rapidly deploying generative AI and agentic systems into frontline workflows: customer support, knowledge management, operations, and decision support. Yet most organizations struggle with three structural challenges:

- **Fragmented User Experience**: AI capabilities are siloed across channels and teams.
- **Opaque Reasoning and Data Flows**: It is difficult to see how models reason, which tools they call, and what data they use.
- **Insufficient Governance**: Security, compliance, cost management, and quality assurance are often bolted on after the fact.

This whitepaper proposes a capability-driven architecture designed to address these challenges from first principles. It combines:

- Power Virtual Agent / Copilot Studio as the **Assist** layer.
- Microsoft Agent Framework (MAF), Semantic Kernel, and LangGraph as the **Orchestrate** layer.
- Azure AI Search, OpenAI embeddings, and ingestion pipelines as the **Retrieve** layer.
- Azure Active Directory / Entra ID, Managed Identities, Key Vault, VNets, App Insights, and policy overlays as the **Govern** layer.

A library of 30 diagrams and patterns describes how these pieces fit together across technical and organizational dimensions.

**GRC Overlay Map for the Agentic AI Control Plane**

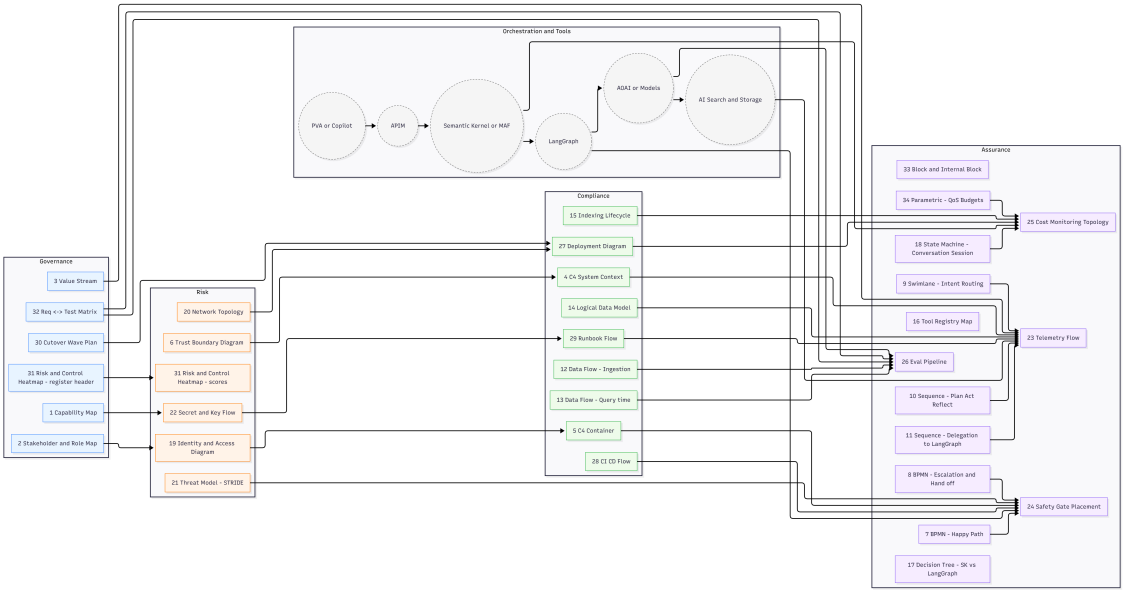



Figure 1: GRC Overlay Map for the Agentic AI Control Plane

The **GRC Overlay Map** presents a complete, systems-engineering view of how the Agentic AI ecosystem—spanning Power Virtual Agent (PVA), Azure API Management (APIM), Semantic Kernel (SK / Microsoft Agent Framework), LangGraph, Azure OpenAI (AOAI), and AI Search/Storage—is governed, secured, and audited. It unifies over thirty architectural and operational artifacts into a four-domain GRC structure that maps every flow, decision, and component to a control objective and accountability boundary.

### 1. Governance Domain

**Purpose**: Establish accountability, oversight, and traceability for AI planning, execution, and user impact.

**Representative artifacts**: Capability Map, Stakeholder Map, Value Stream, Cutover Plan, Risk/Heatmap Register, Req $\leftrightarrow$ Test Matrix.

- The **Capability Map** identifies functional control surfaces—Assist, Orchestrate, Retrieve, Govern—as the foundation for policy domains (for example, model governance, data governance, tool governance).
- The **Stakeholder Map** ties PVA authors, SK tool owners, data stewards, and SecOps teams to explicit decision rights under a RACI framework.
- The **Value Stream** provides end-to-end visibility from user intent $\rightarrow$ orchestration $\rightarrow$ grounded answer $\rightarrow$ feedback loop, enabling business continuity and ethical oversight.

- **Cutover & Risk Heatmaps** link each migration or iteration to governance checkpoints and tolerance thresholds.
- The **Req ↔ Test Matrix** anchors requirements, controls, and metrics to measurable outcomes (accuracy, latency, $/turn).

Together, these diagrams define the "who governs what and how" backbone of the entire system.

#### 2. Risk Domain

**Purpose**: Identify and mitigate technical, operational, and ethical risks inherent in multi-agent orchestration.

**Representative artifacts**: Trust Boundaries, Identity & Access, Network Topology, Threat Model (STRIDE), Secret/Key Flows, Risk Heatmaps.

- The **Trust Boundary Diagram** defines isolation zones (Public, DMZ, Private VNet/Private Endpoint), ensuring least-privilege data exposure across PVA, orchestrators, and LangGraph.
- The **Identity & Access Diagram** captures relationships between Azure AD, Managed Identities, and Key Vault roles, ensuring traceable, auditable access paths.
- The **Threat Model** (DFD/STRIDE) quantifies risks such as prompt injection, data exfiltration, or unauthorized tool execution, assigning controls like token limits, validation filters, and scope enforcement.
- **Secret & Key Flow** diagrams specify who retrieves what, from where, under which identity, aligning to SOC 2 and ISO 27001 access policies.

These diagrams create a living risk register that can be tied directly to mitigation tasks, CI/CD policies, and audit evidence.

## 2. Capability Map: Assist, Orchestrate, Retrieve, Govern

### 2.1 The Four Foundational Pillars

The **AI Capability Map – Assist, Orchestrate, Retrieve, Govern** (standard and "Bright Theme" variants) defines the conceptual backbone of the architecture.

#### Assist (Light Blue).

- Frontline experience surfaces: Teams, web chat, email, IVR.
- Conversation management via Power Virtual Agent / Copilot Studio.
- Responsibilities: intent recognition, slot-filling, hand-off to orchestrator, optional human escalation.

**Orchestrate (Yellow-Orange).**

- Central "brain" implemented with Microsoft Agent Framework, Semantic Kernel, and LangGraph.
- Responsibilities: plan–act–reflect loops, tool selection and invocation, multi-step workflows, error handling.

**Retrieve (Green).**

- Knowledge and grounding via Azure AI Search, hybrid vector retrieval, and OpenAI embeddings.
- Responsibilities: ingestion pipelines, RAG, grounding packet assembly, citations and evidence.

**Govern (Magenta).**

- Trust and control plane: AAD/MSI, VNets, Key Vault, App Insights, safety filters, cost and evaluation pipelines.
- Responsibilities: identity and access control, network security, observability, quality assurance, and financial governance.

The arrows between these pillars model runtime data flow (user input $\rightarrow$ orchestration $\rightarrow$ retrieval $\rightarrow$ governed output), while dashed lines represent governance overlays that supervise security, monitoring, evaluation, and cost at every step.

This capability map is not just a diagram—it is a design contract that every component and team must align to.

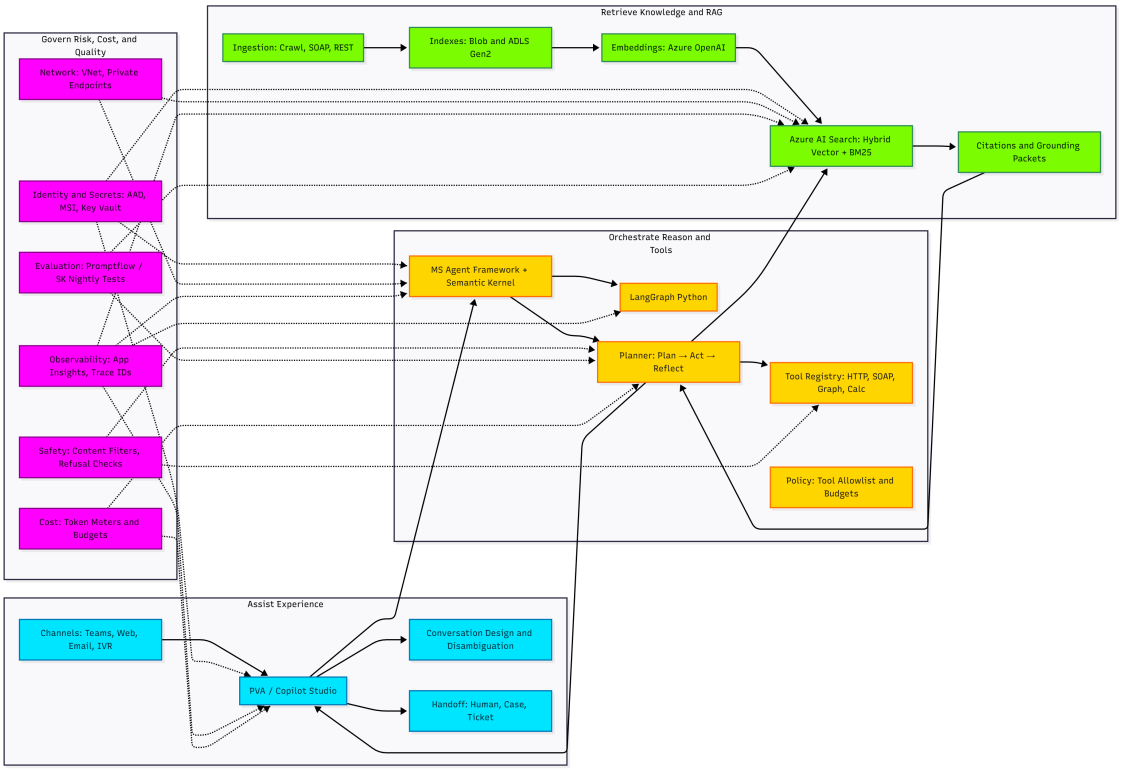



Figure 2: Pattern 1: AI Capability Map – Assist, Orchestrate, Retrieve, Govern

## 3. Stakeholders, Roles, and Value Stream

### 3.1 Stakeholder & Role Map

The **Stakeholder & Role Map** aligns human responsibilities to the four pillars:

**PVA Authors & Conversation Designers (Assist)** Build and maintain topics, dialog flows, and escalation rules.

**Orchestrator Engineers (Orchestrate)** Own Semantic Kernel planners, MAF agents, LangGraph workflows, and tool adapters.

**Data Owners & Knowledge Curators (Retrieve)** Govern ingestion connectors, document sources, schemas, and index quality.

**SecOps / GRC / Compliance (Govern)** Oversee identity, audit, threat model, policy enforcement, and regulatory alignment.

**Finance / FinOps (Govern)** Define and enforce budgets, SLO/SLA overlays, and cost monitoring policies.

**Product & Business Owners (Cross-cutting)** Define intents, KPIs, and success metrics; prioritize use cases.

Dashed connections in the stakeholder map highlight cross-cutting roles such as SecOps and Finance, which influence all pillars. This alignment prevents ownership gaps and supports clear accountability for operational incidents and governance decisions.

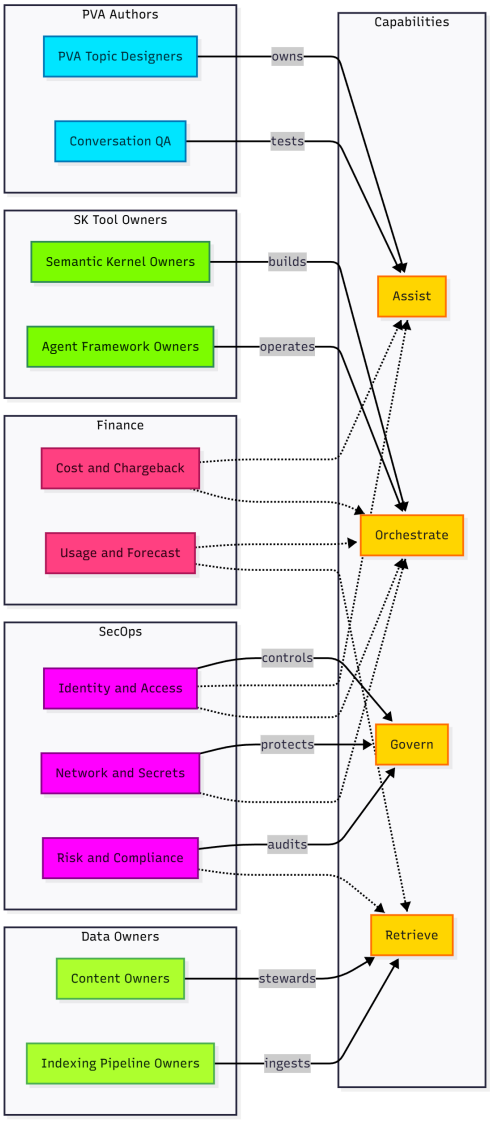



Figure 3: Pattern 3: Stakeholder & Role Map — Aligning Responsibilities Across the AI Capability Stack

### 3.2 AI Value Stream: From Question to Improvement

The **AI Value Stream** describes the end-to-end lifecycle of an interaction:

- **User Layer** – A user asks a question or initiates a task over Teams, web, email, or IVR.
- **Assist Layer** – PVA/Copilot captures the message, identifies the topic or intent, and constructs a structured payload.

- **Orchestrate Layer** – The orchestrator (MAF/SK/LangGraph) plans the interaction, calling tools and retrieval services as needed.
- **Retrieve Layer** – The RAG pipeline performs hybrid search, assembles grounding packets, and returns evidence to the orchestrator.
- **Govern Layer** – Safety filters, identity and access checks, telemetry, and cost tracking run alongside each turn.
- **Evaluation Loop** – Offline evaluation (e.g., Promptflow/SK jobs) reviews traces for factuality, citation coverage, safety, latency, and user satisfaction.

Every interaction generates data points for improvement: logs, traces, cost metrics, and evaluation scores. These feed back into prompt templates, index configuration, tool policies, and safety rules, enabling continuous optimization.

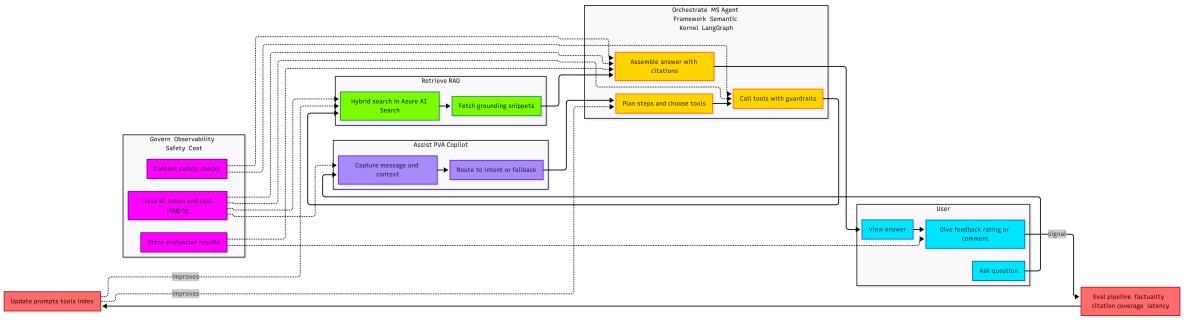



Figure 4: Pattern 4: AI Value Stream — From User Question to Grounded Answer and Continuous Improvement

## 4. System Architecture: Context, Containers, and Trust Boundaries

### 4.1 System Context

The **C4 System Context Diagram** describes the outer boundary of the AI system:

**Public Zone.**

- Microsoft Teams, web clients, email gateways, telephony / IVR.
- These channels communicate with PVA or Copilot Studio.

**Azure Entry Zone.**

- Azure API Management (APIM) as the secure ingress point.
- Optional Azure Front Door / Application Gateway + WAF.

**Private VNet – Orchestration Zone.**

- Orchestrator services (MAF/SK), LangGraph workers.
- Azure Functions and internal tool adapters.
- Identity and secret management via Entra ID (AAD) and Key Vault.

**Data & AI Services Zone.**

- Azure OpenAI, Azure AI Search, Storage (Blob/ADLS), observability tools (App Insights, Log Analytics).

**Governance & Observability.**

- Cross-cutting services for telemetry, security analytics, and compliance monitoring.

This view ensures that the system is understood as a cohesive product, not a loose collection of microservices.

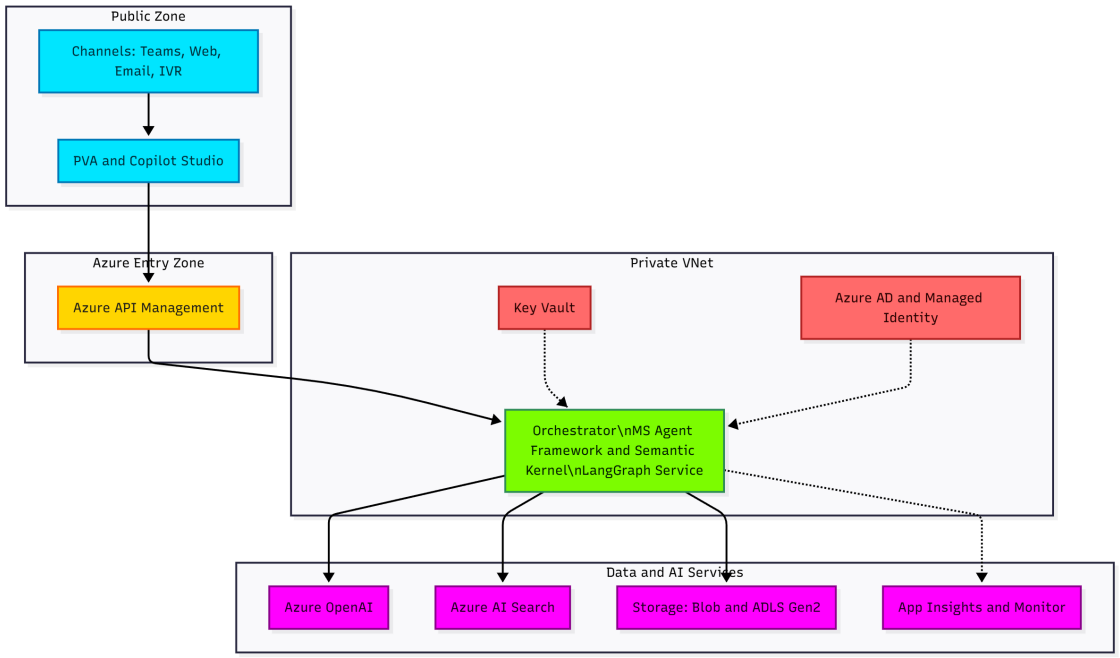



Figure 5: Pattern 5: C4: System Context — End-to-End AI Architecture Overview

### 4.2 Container-Level Decomposition

The **C4 Container Diagram** refines the context into deployable components:

**Assist Containers.**

- PVA skill apps, web adapters, Teams bots, IVR connectors.

**Entry Containers.**

- APIM APIs and policies (throttling, auth, logging, transformation).

**Orchestrate Containers.**

- MAF/SK Service (C# or Python-based orchestrator).
- LangGraph Service (Python-based multi-agent engine).
- Azure Functions for long-running tasks and integration-specific tools.

**Retrieve Containers.**

- AI Search instances, embedding services, data access APIs.

**Governance Containers.**

- Observability agents, security monitoring, cost aggregation, evaluation pipelines.

This decomposition is the blueprint for deployment and scaling strategies, making it possible to assign performance and reliability targets per container.

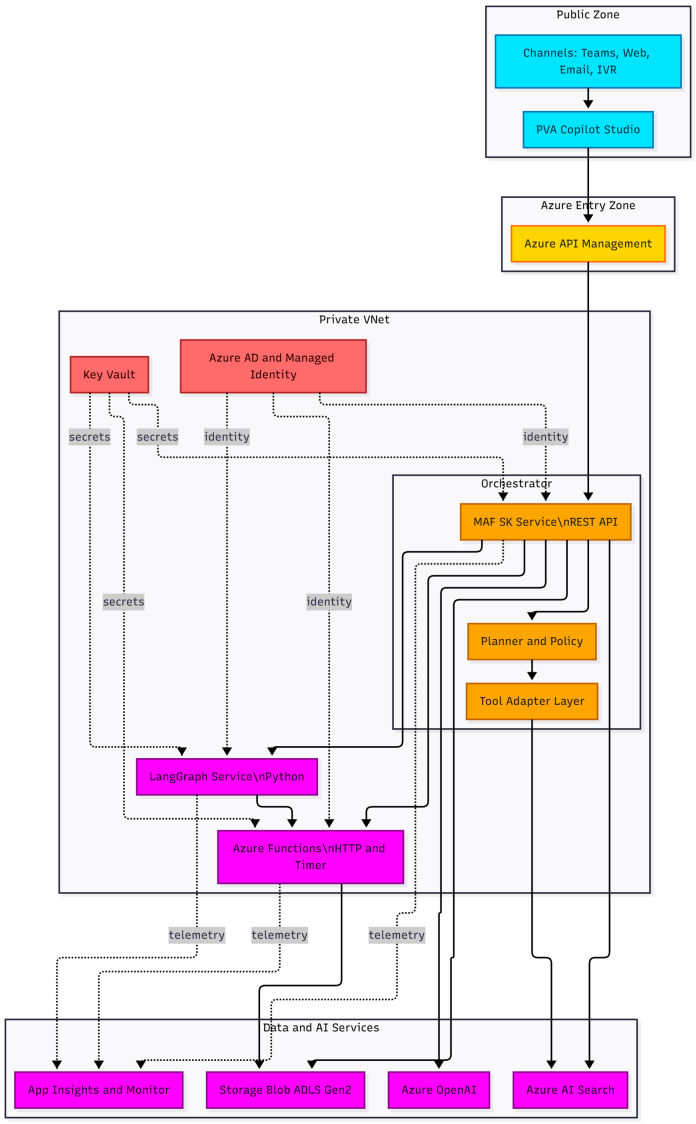



Figure 6: Pattern 6: C4: Container Diagram — Functional Decomposition of the AI Orchestration Layer

### 4.3 Trust Boundaries

The **Trust Boundary Diagram** defines where security controls must be enforced:

- **Public Zone → DMZ/Edge Zone**: TLS termination, WAF, rate limiting.
- **DMZ/Edge Zone → Private App Zone (VNet)**: APIM policy enforcement, IP restrictions, mutual TLS, identity-based access.
- **Private App Zone → Provider Tenant Services** (Azure OpenAI, AI Search, Storage): private endpoints, managed identities, RBAC, network isolation.

By explicitly modeling Public, DMZ, Private, and Provider zones, the architecture enables zero-trust patterns: no implicit trust based on network location; identity and least privilege are enforced throughout.

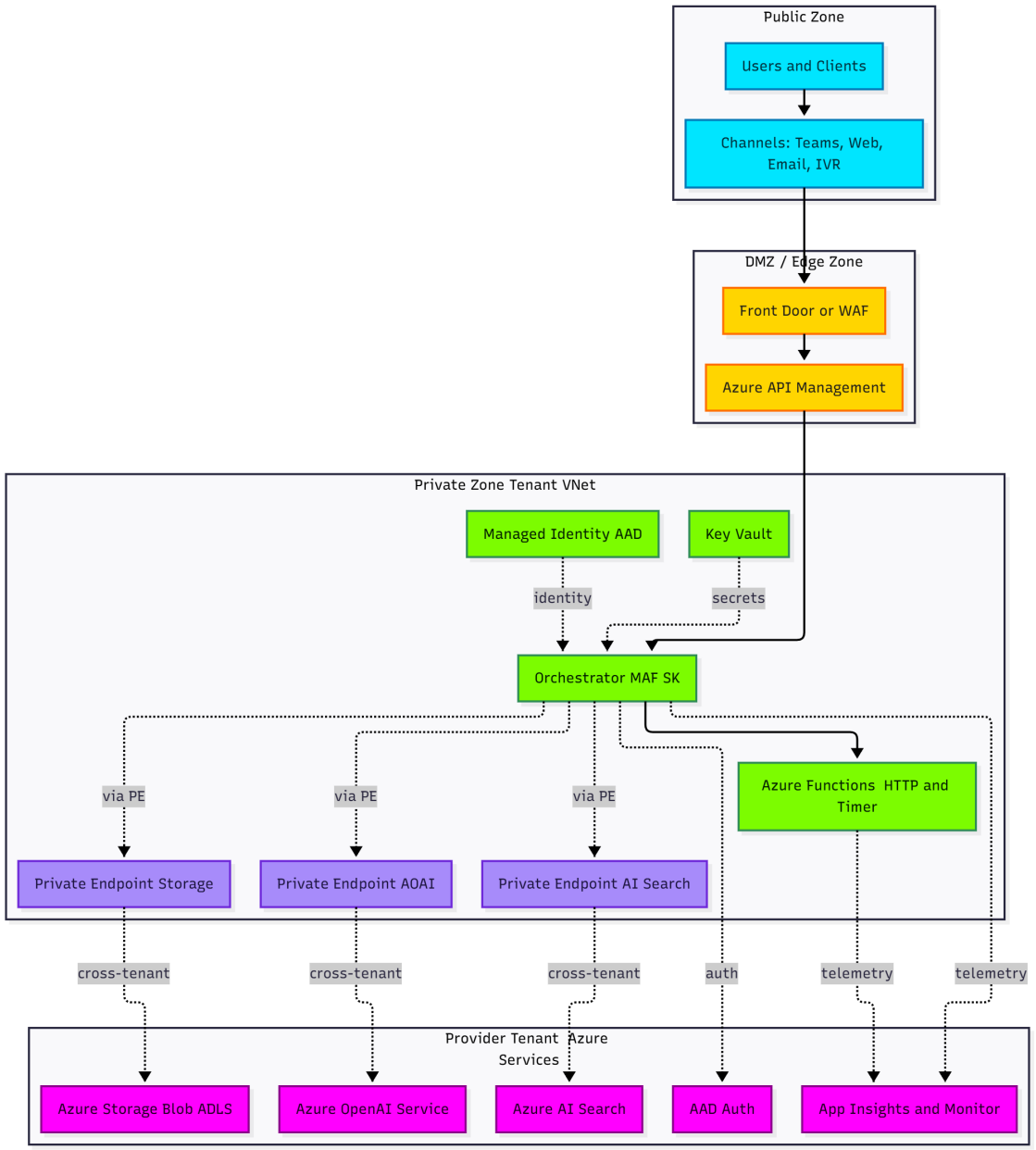



Figure 7: Pattern 7: Trust Boundary Diagram — Public, DMZ, Private (VNet/Private Endpoints), and Cross-Tenant Edges

## 5. Runtime Behavior: Conversation, Intent Routing, and Agentic Loops

### 5.1 Conversation Happy Path

The **BPMN Conversation Happy Path** maps a single turn across swimlanes:

- **User** – initiates and receives responses.
- **Assist (PVA/Copilot)** – captures intent, prepares payload.
- **Entry (APIM)** – authenticates, logs, adds trace identifiers.
- **Orchestrate (MAF/SK)** – plans actions, invokes tools and RAG.
- **Retrieve (RAG/Azure AI Search)** – fetches evidence.
- **Govern** – safety checks, telemetry, and cost tracking.

The result is a grounded, citation-rich answer returned to PVA. This path is the target for performance tuning and SLOs (latency, accuracy, and reliability).

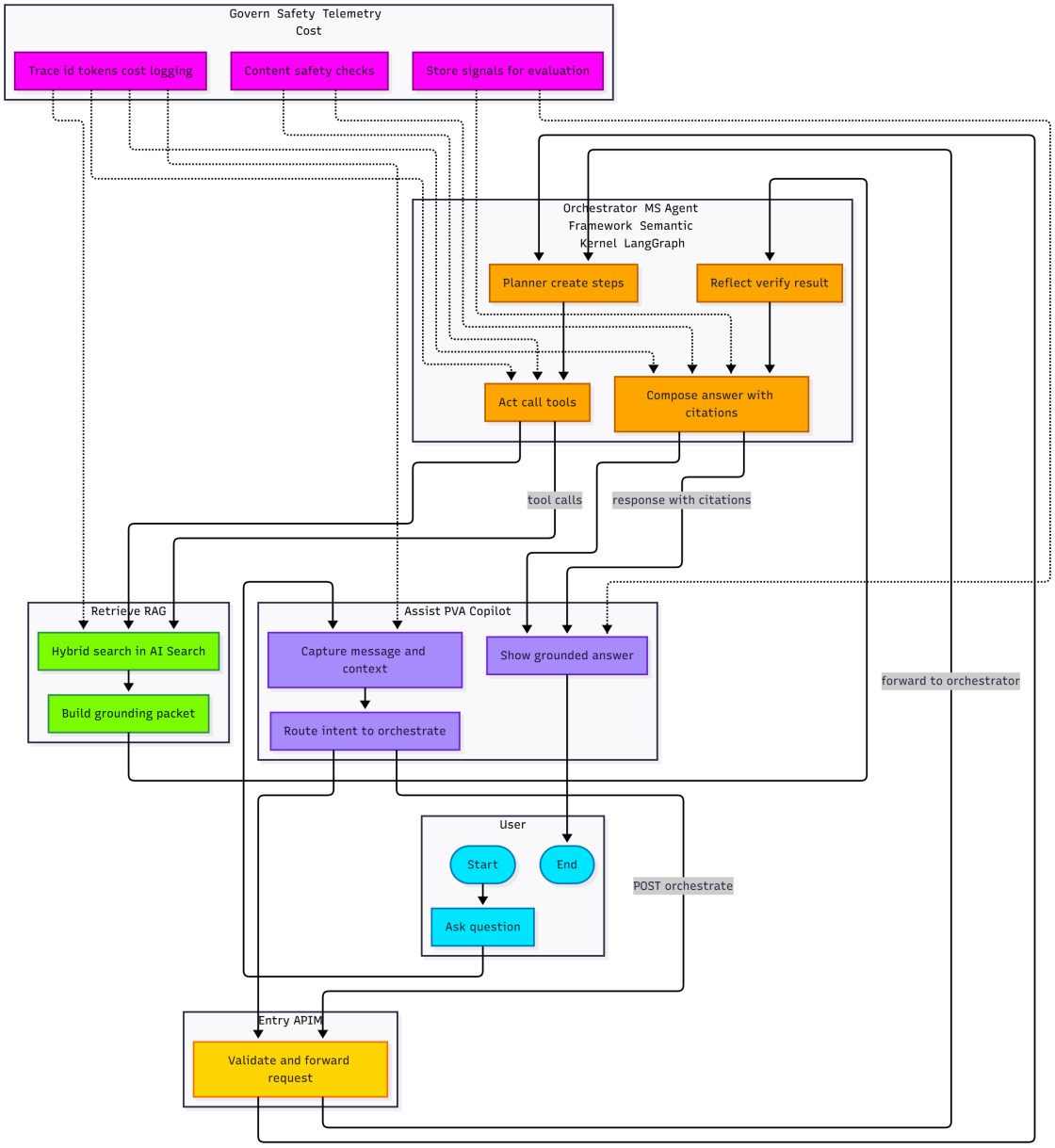



Figure 8: Pattern 8: BPMN: Conversation Happy Path — Intent Routing to Grounded AI Answer

### 5.2 Escalation & Hand-Off

When confidence in a generated answer is low or safety concerns arise, the **Escalation & Hand-Off BPMN** defines the alternative path:

- Confidence score falls below threshold.
- Orchestrator invokes Human Agent and Ticketing lanes.
- A ticket is created and routed to the right queue.

- Callback or follow-up is scheduled; SLAs and timers are applied.
- All events are logged for evaluation and continuous improvement.

This ensures that automation is bounded by human oversight, preventing the system from giving unreliable answers in high-risk scenarios.

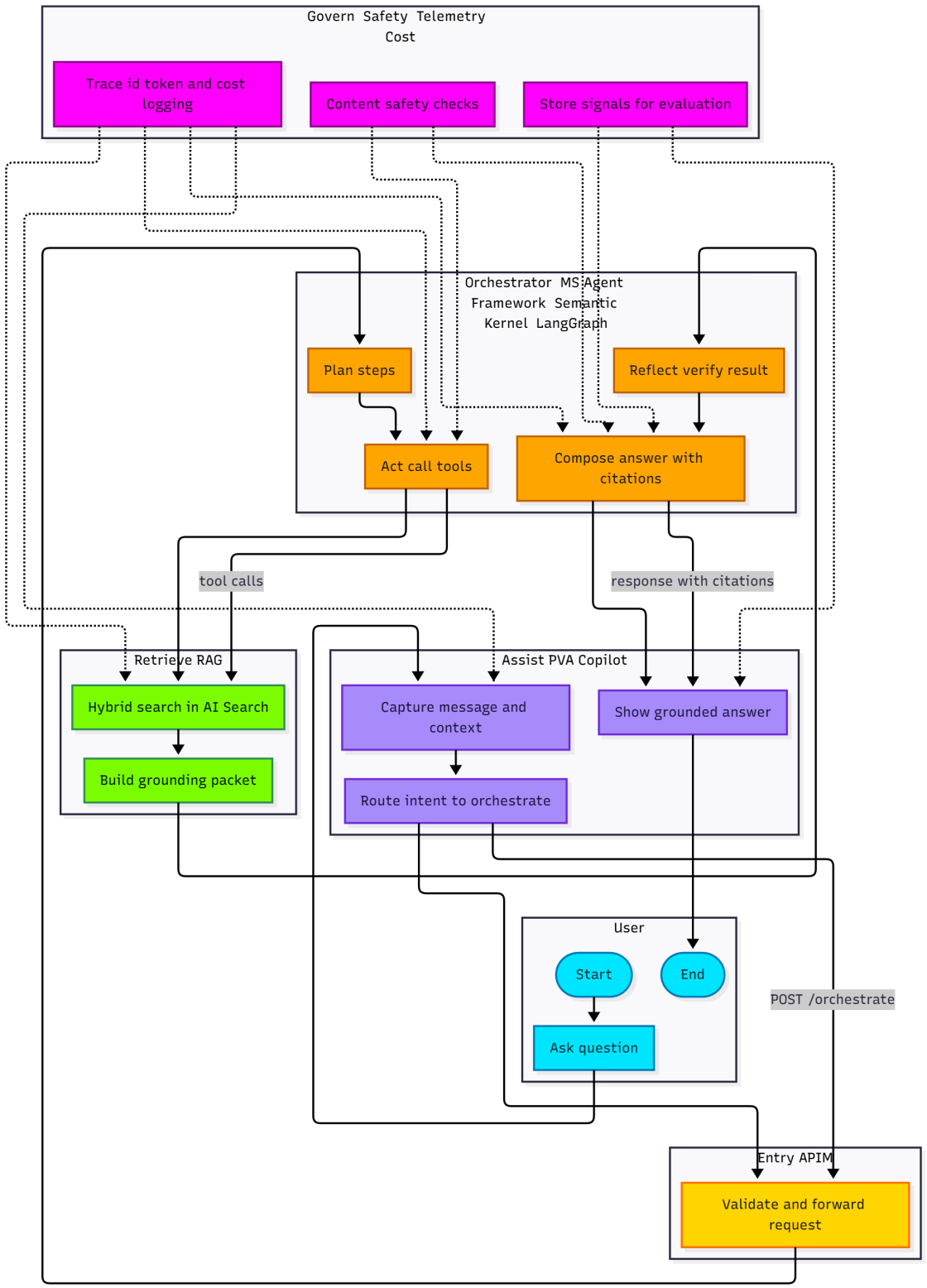



Figure 9: Pattern 9: BPMN: Escalation & Hand-Off — Managing Low Confidence AI Responses

### 5.3 Intent Routing

The **Intent Routing Diagram** details how requests move from PVA to orchestrators and RAG:

- PVA classifies or infers intent and builds a structured action payload.
- APIM authenticates, logs, and routes the request.
- Semantic Kernel's planner maps intent to a plan of tools and retrieval steps.
- Tools (HTTP, SOAP, Graph, Calc, custom functions) are invoked where appropriate.
- LangGraph is used when multi-step, branching workflows or multi-agent patterns are needed.
- RAG is triggered to fetch data via Azure AI Search and embeddings when grounding is required.

This pattern ensures clear separation of concerns between channel, planning, tool execution, and knowledge retrieval.

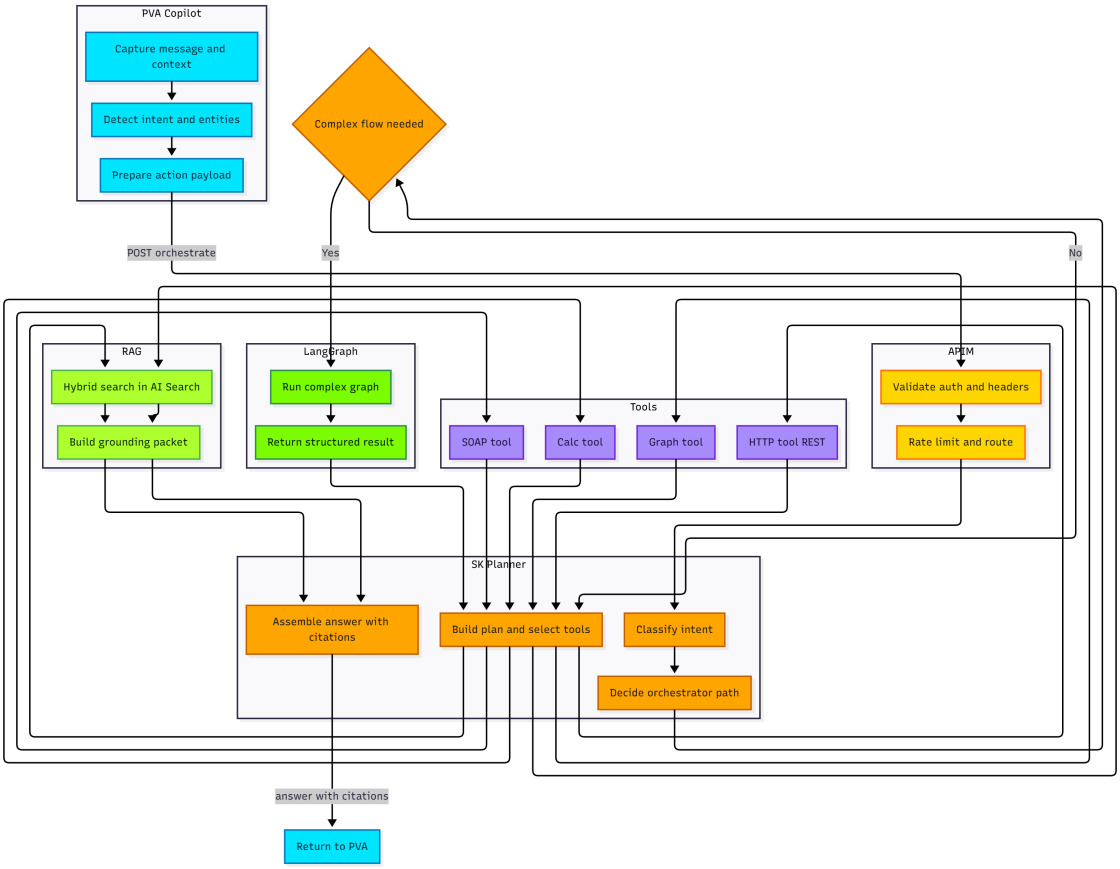



Figure 10: Pattern 10: Intent Routing — From PVA to LangGraph and RAG Orchestration

### 5.4 Plan–Act–Reflect Turn Cycle

At the micro level, the orchestrator follows a **Plan–Act–Reflect** loop:

- **Plan** – Interpret the prompt, context, and history. Decide which tools and retrieval steps to take.
- **Act** – Execute the plan: invoke tools, run RAG, call models; collect intermediate results and telemetry.
- **Reflect** – Evaluate outcomes for correctness, safety, and completeness. Decide whether to refine the plan, escalate, or finalize.

This loop can be implemented using Semantic Kernel planners and/or LangGraph workflows, with reflection steps potentially using meta-prompts or evaluation functions. It provides the agentic behavior needed for complex tasks while preserving control.

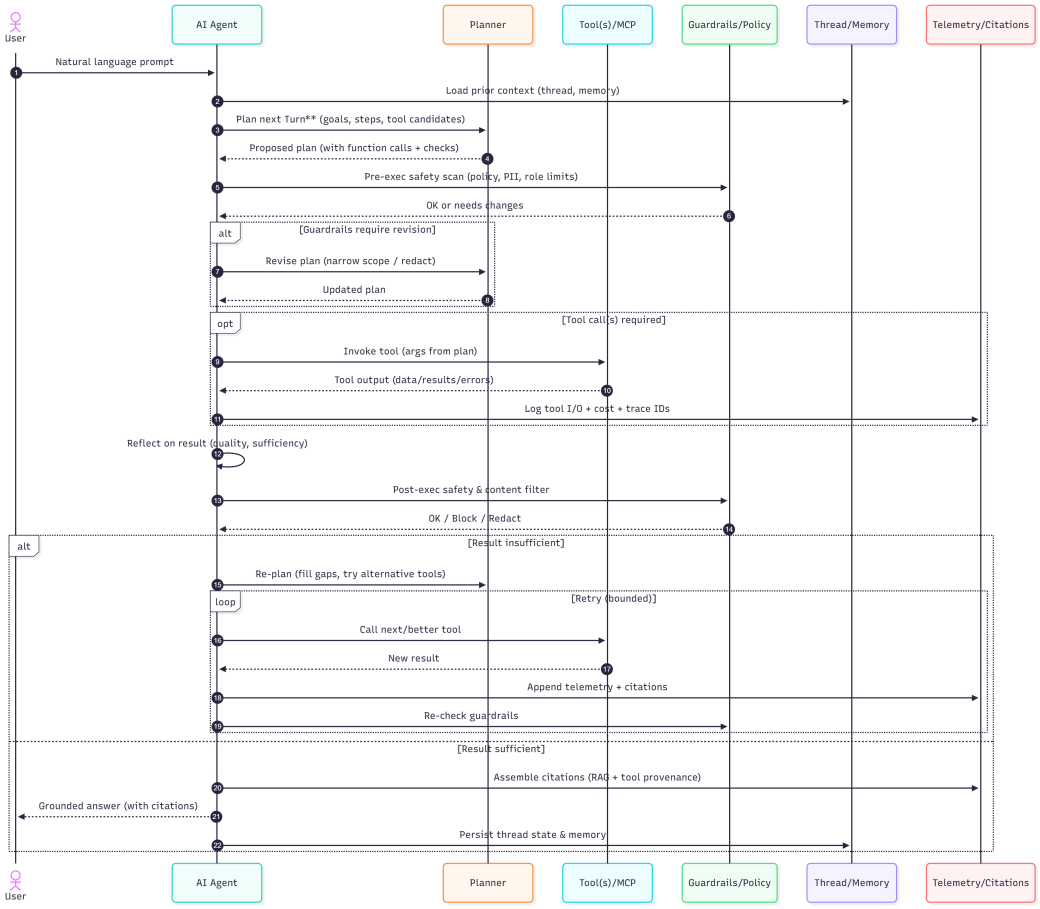



Figure 11: Pattern 11: Plan–Act–Reflect Turn Sequence: A Closed-Loop Agentic Interaction Model

### 5.5 Conversation State Machine

A **State Machine** formalizes session lifecycle:

Started → Routed → Tooled → Grounded → Answered / Escalated

with global transitions: *Timeout*, *Abort*, *Retry*.

Each transition emits telemetry under a unified `trace_id`, making it possible to reconstruct and audit entire sessions for compliance and debugging.

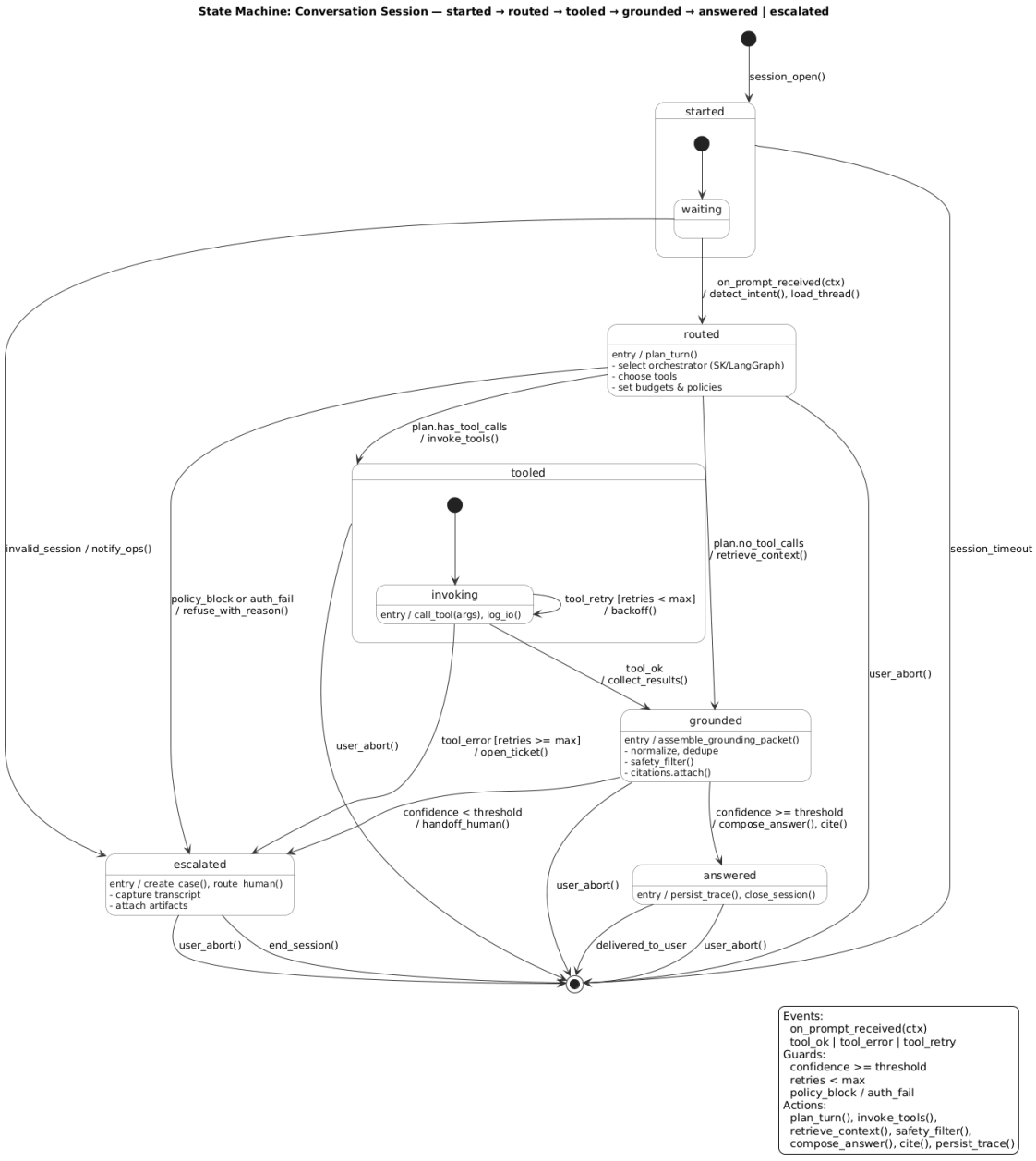



Figure 12: Pattern 18: State Machine: Conversation Session — Lifecycle from Start to Escalation in an Agentic AI Orchestrator

## 6. Data, Knowledge, and Tools

### 6.1 Ingestion Data Flow

The **Ingestion Data Flow** defines how raw enterprise data becomes RAG-ready:

- **Sources** – SharePoint, web, file shares, SaaS systems, line-of-business apps.
- **Ingestion** – Connectors, crawlers, APIs pull data at controlled intervals.
- **Processing** – Parsing, normalization, deduplication, redaction, metadata enrichment.
- **Chunking** – Splitting into semantic chunks with overlaps.
- **Embedding** – Converting chunks into vector representations.
- **Indexing** – Writing chunks and vectors into Azure AI Search.
- **Observability & Governance** – Lineage tracking, PII controls, audit logs.

This pipeline underpins the **Retrieve** pillar and is essential for reducing hallucination risk and ensuring up-to-date knowledge.

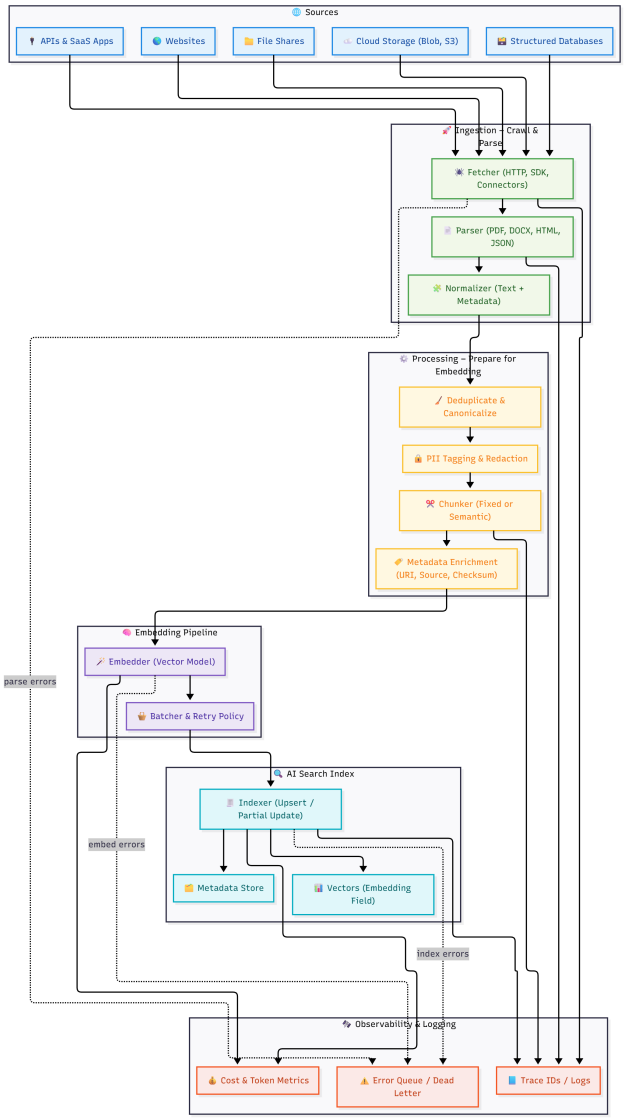



Figure 13: Pattern 13: Data Flow (Ingestion) — From Raw Sources to AI Search Index via Crawl, Parse, Chunk, Embed, and Upsert

### 6.2 Query-Time Data Flow

The **Query-Time Data Flow** explains how user queries are grounded:

- **Query Input** – Orchestrator receives the user's question and context.
- **Query Enrichment** – Optional expansion, filters, or rephrasing.
- **Hybrid Retrieval** – BM25 keyword + vector similarity, optional reranking.
- **Grounding Packet Assembly** – Deduplicated, filtered, and citation-tagged snippets.
- **Model Inference** – System prompts + user query + grounding packet.

- **Post-Processing** – Safety checks, formatting, confidence scoring, refusal logic.
- **Observability** – Trace IDs, cost metrics, and error handling.

This is the core RAG loop, ensuring that answers are tied to evidence and are traceable.

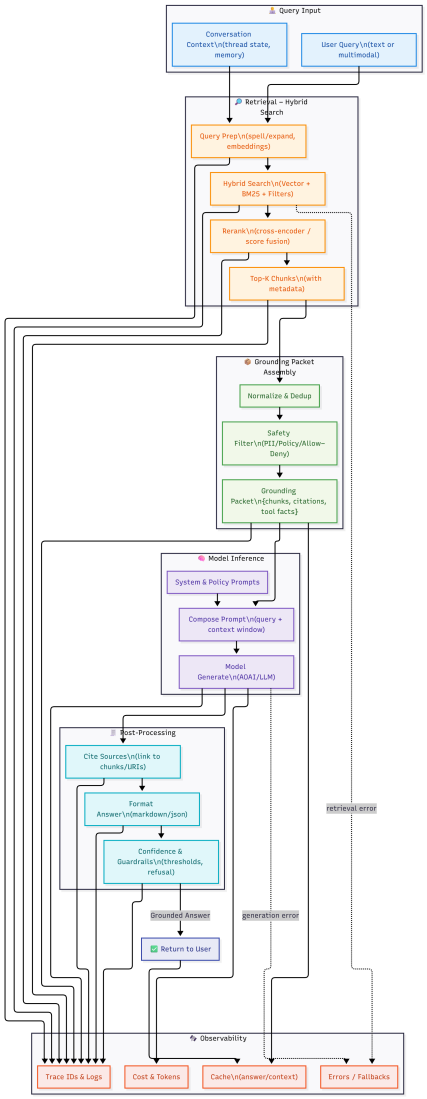



Figure 14: Pattern 14: Data Flow (Query-Time) — Query → Hybrid Search → Grounding Packet → Model → Cite

### 6.3 Logical Data Model

The **Logical Data Model** defines shared entities:

- **Document, Chunk, Embedding** – content and its vector representation.
- **Citation** – mapping between answer spans and the source chunks.

- **Trace** – end-to-end record of an interaction or job execution.
- **ToolCall** – metadata about tool invocations (name, version, input, output).

These entities support lineage, observability, and compliance, and are reused across ingestion, query-time, evaluation, and cost analysis.

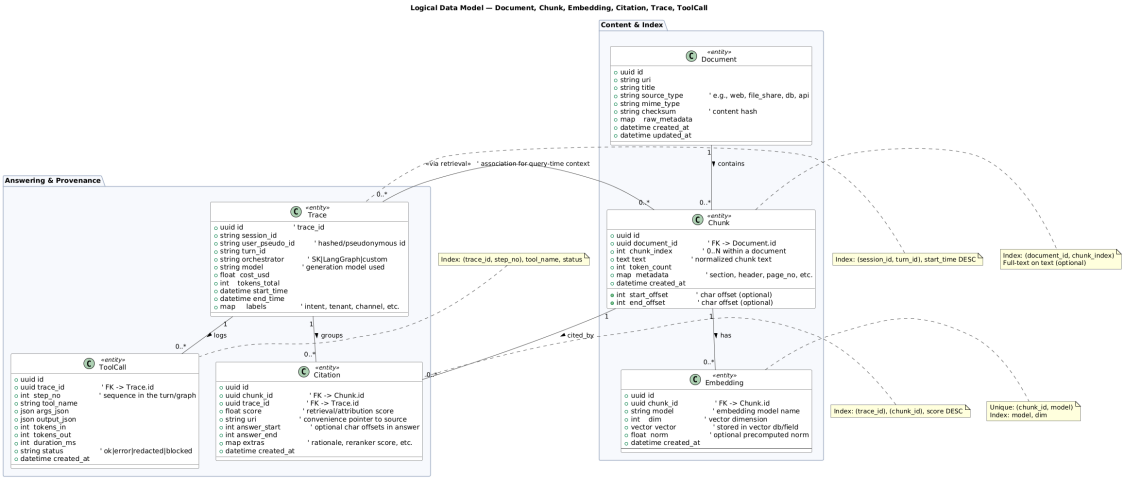



Figure 15: Pattern 15: Logical Data Model — Document, Chunk, Embedding, Citation, Trace, and ToolCall Entities

### 6.4 Tool Registry and Orchestration Choice

The **Tool Registry Map** is a central catalog for Semantic Kernel tools and LangGraph parity tools, enriched with:

- Schemas, versions, policies, budgets, and observability hooks.

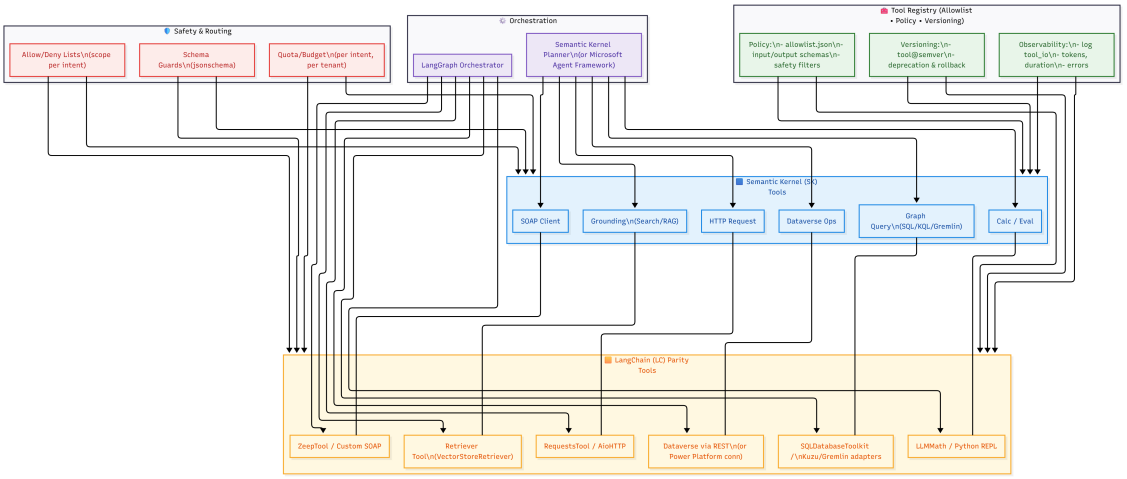



Figure 16: Pattern 16: Tool Registry Map — Semantic Kernel Tools and LangChain Parity Tools with Policy Governance

The **SK vs LangGraph Decision Tree** provides guidelines for choosing:

- **SK** when the environment is Microsoft-first, workflows are relatively linear, and Azure-native governance is paramount.
- **LangGraph** when Python ecosystems, research-style workflows, or complex multi-agent flows dominate.
- A **hybrid** approach when both are required and can be constrained under a common governance model.

Together, these patterns ensure that tools are visible, controlled, and deployed in the right orchestration context.

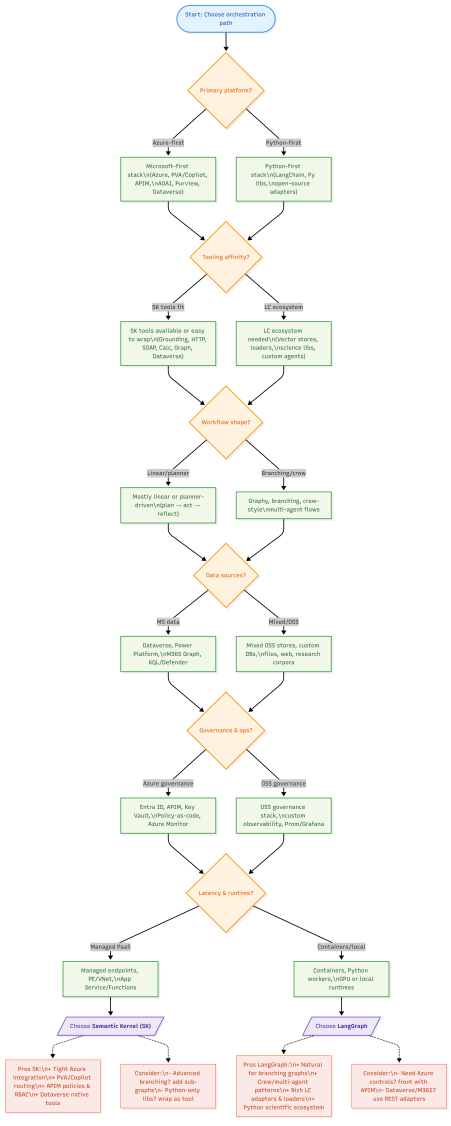



Figure 17: Pattern 17: Decision Tree: Semantic Kernel vs LangGraph — Choosing Between Microsoft-First and Pythonic Multi-Agent Orchestration

## 7. Governance, Risk, Compliance, and Assurance

### 7.1 GRC Overlay Map

The **GRC Overlay** is a meta-view that binds all technical artifacts into four domains:

- **Governance** – Policies, roles, control objectives.
- **Risk** – Threat modeling, risk registers, mitigations.
- **Compliance** – Mappings to NIST, ISO, SOC 2, HIPAA, EU AI considerations.
- **Assurance** – Testing, evaluation pipelines, audits, dashboard reporting.

The Orchestration Layer runs through all four domains, ensuring that agentic behavior is never isolated from governance.

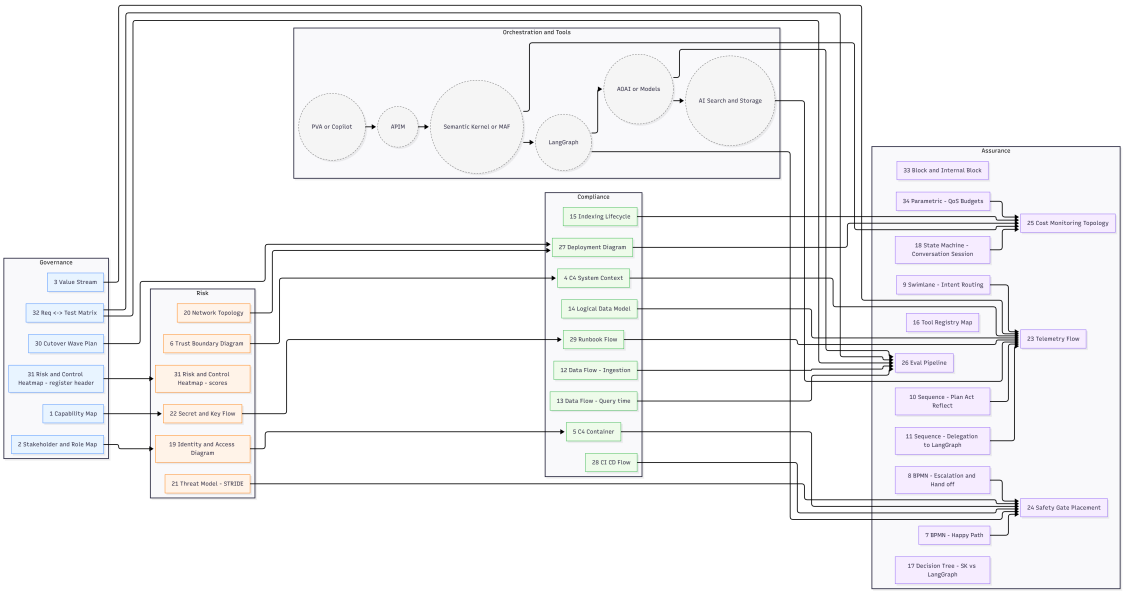



Figure 18: Pattern 12: GRC Overlay Map for the Agentic AI Control Plane — A Unified Framework for Governance, Risk, Compliance, and Assurance

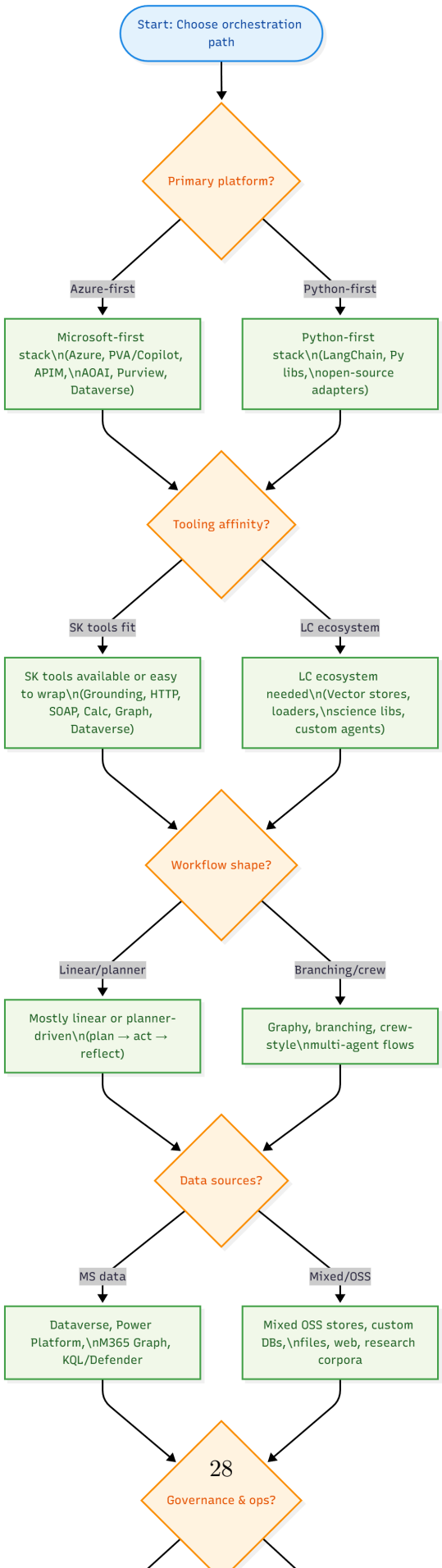
### 7.2 Identity, Network, and Threat Modeling

**Identity & Access Diagram.** Entra ID (AAD), Managed Identities, and Key Vault define identity and secrets management in a multi-tenant environment.

**Network Topology.** Hub–spoke VNets, subnets, private endpoints, and APIM control ingress and egress.

**Threat Model (DFD).** STRIDE-based analysis across Public, DMZ, Private, Data, Model, and Observability zones, with explicit mitigations for:

- Prompt injection.
- Tool misuse.
- Data poisoning.
- Transport and identity threats.
- Auditability and resilience.


Start: Choose orchestration path
Primary platform?
Azure-first
Python-first
Microsoft-first stack\n(Azure, PVA/Copilot, APIM,\nAOAI, Purview, Dataverse)
Python-first stack\n(LangChain, Py libs,\nopen-source adapters)
Tooling affinity?
SK tools fit
LC ecosystem
SK tools available or easy to wrap\n(Grounding, HTTP, SOAP, Calc, Graph, Dataverse)
LC ecosystem needed\n(Vector stores, loaders,\nscience libs, custom agents)
Workflow shape?
Linear/planner
Branching/crew
Mostly linear or planner-driven\n(plan → act → reflect)
Graphy, branching, crew-style\nmulti-agent flows
Data sources?
MS data
Mixed/OSS
Dataverse, Power Platform,\nM365 Graph, KQL/Defender
Mixed OSS stores, custom DBs,\nfiles, web, research corpora
Governance & ops?

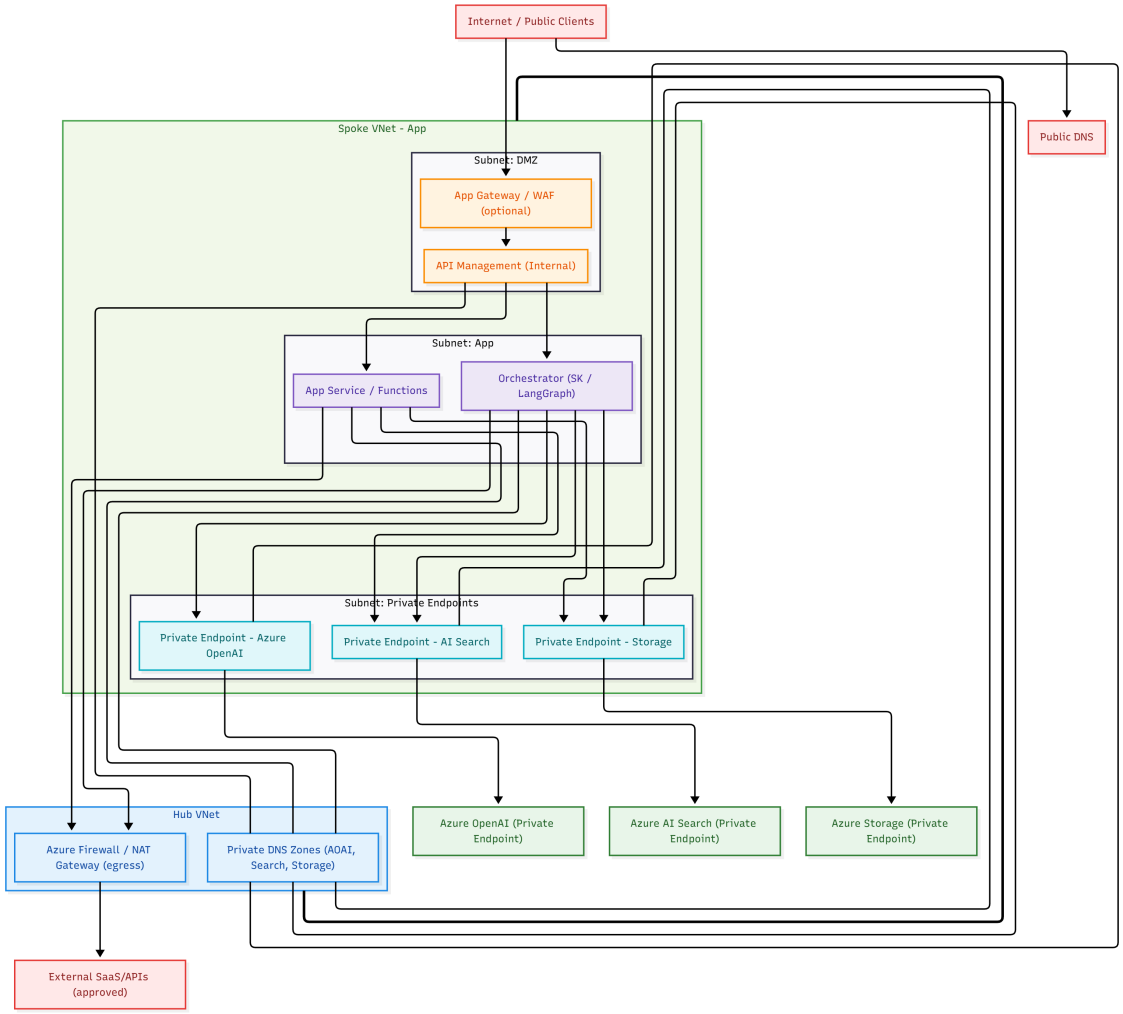



Figure 20: Pattern 20: Network Topology — VNets, Subnets, Private Endpoints, APIM, and Egress Flow

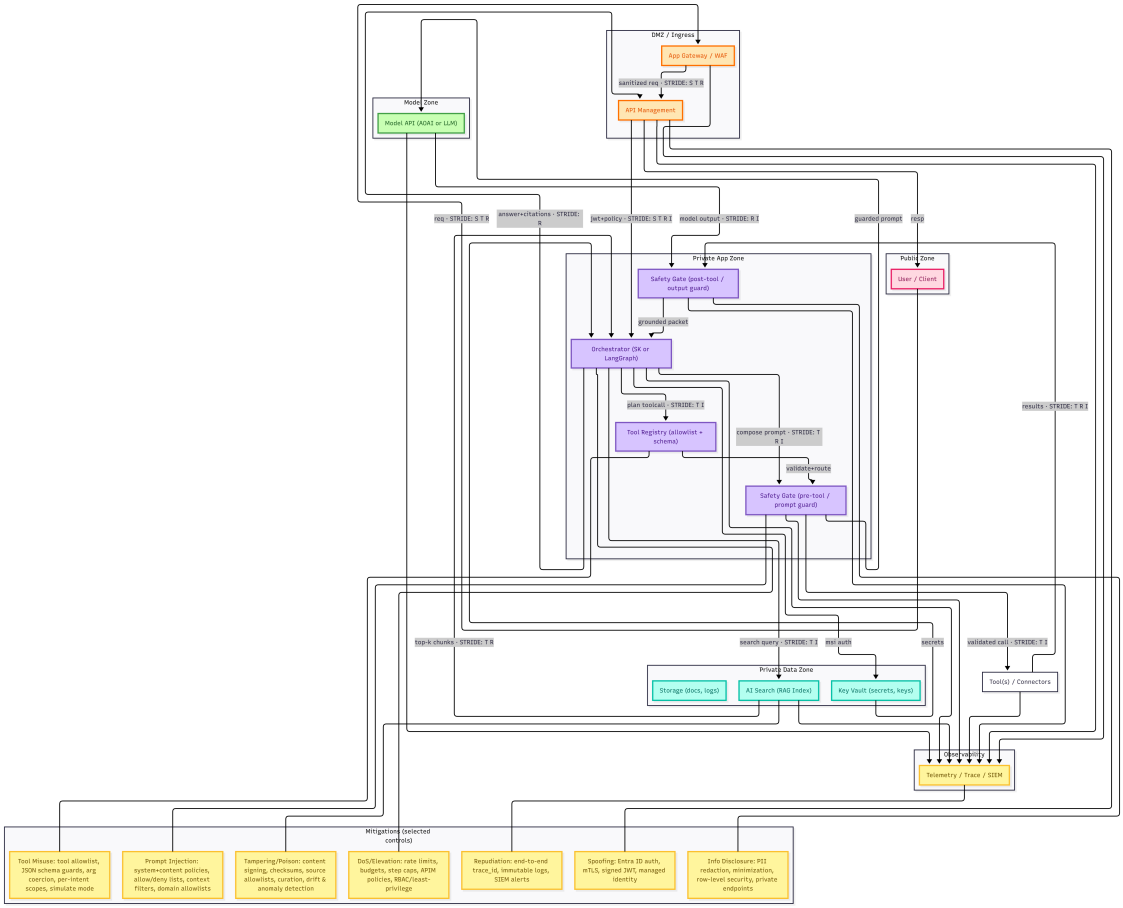



Figure 21: Pattern 21: Threat Model (DFD) — Bright, ELK-Layout View of STRIDE Risks & Mitigations for Prompt Injection and Tool Misuse

### 7.3 Secrets, Telemetry, Safety, and Cost

**Secret & Key Flow.** Zero-trust secret management with no static keys, private endpoints, RBAC, and audit logs.

**Telemetry Flow.** `trace_id` propagation from PVA → APIM → orchestrator → tools → monitoring, enabling full traceability.

**Safety Gate Placement.** Pre-tool and post-tool safety filters, allow/deny lists, and structured refusal/escalation paths.

**Cost Monitoring Topology.** Token meters and per-tool/per-intent budgets, integrated with observability and alerting.

These patterns ensure that the **Govern** pillar is fully realized in code, infrastructure, and process.

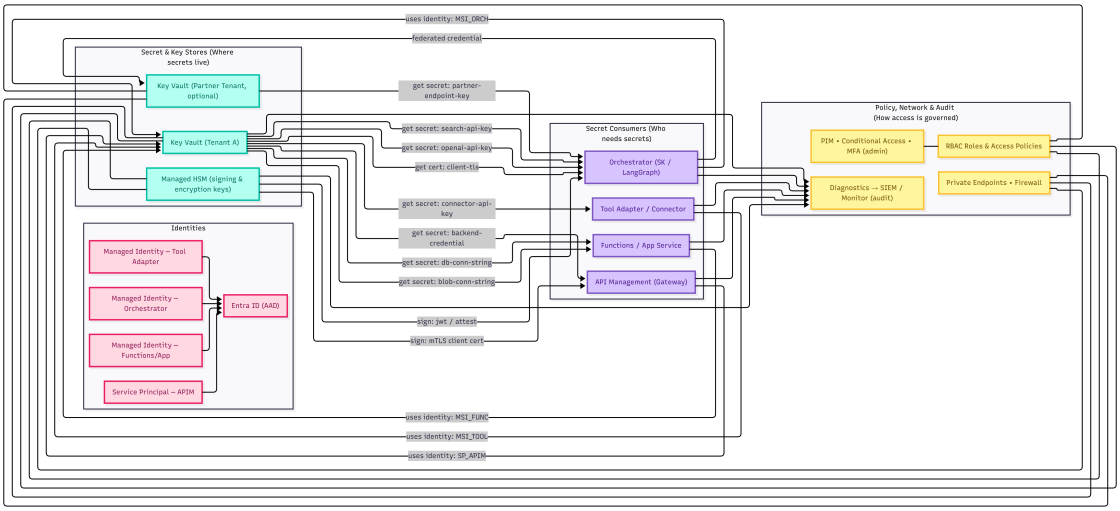



Figure 22: Pattern 22: Secret & Key Flow — Identities, Retrieval Paths, and Governance for Secure Secret Management

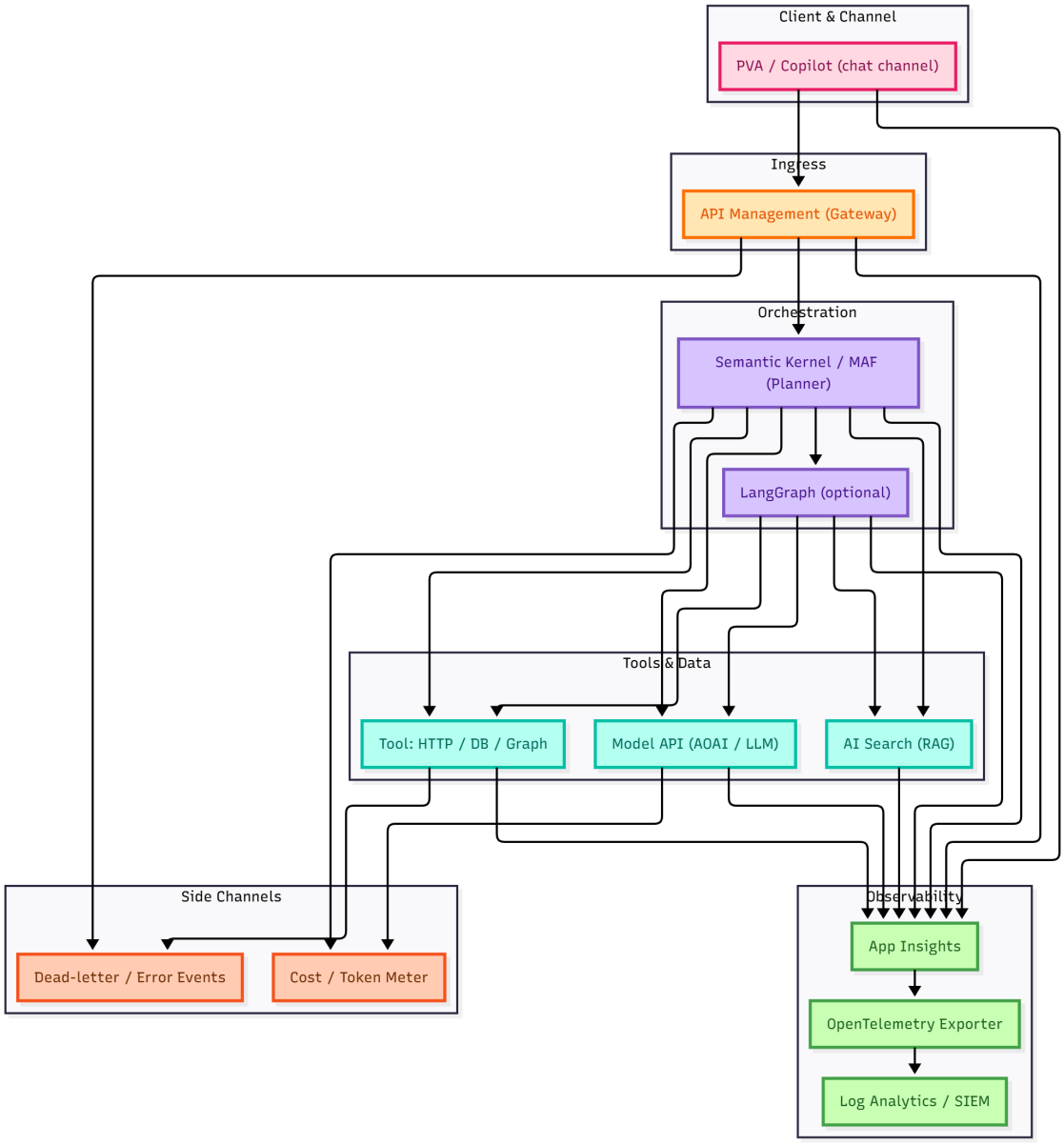
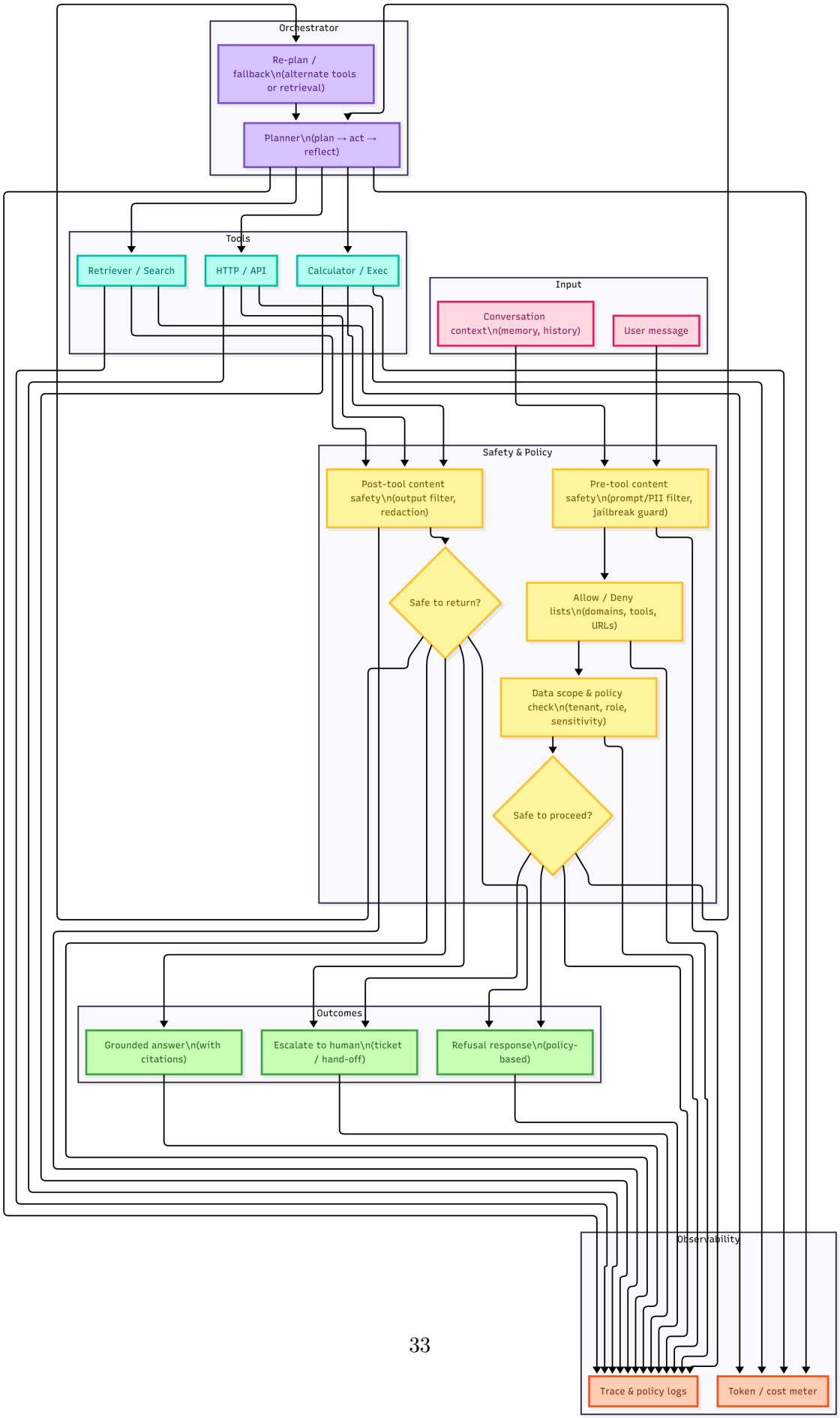



Figure 23: Pattern 23: Telemetry Flow — trace_id Propagation Across PVA → APIM → Semantic Kernel → Tools → Insights


Orchestrator
Re-plan / fallback\n(alternate tools or retrieval)
Planner\n(plan → act → reflect)
Tools
Retriever / Search
HTTP / API
Calculator / Exec
Input
Conversation context\n(memory, history)
User message
Safety & Policy
Post-tool content safety\n(output filter, redaction)
Pre-tool content safety\n(prompt/PII filter, jailbreak guard)
Safe to return?
Allow / Deny lists\n(domains, tools, URLs)
Data scope & policy check\n(tenant, role, sensitivity)
Safe to proceed?
Outcomes
Grounded answer\n(with citations)
Escalate to human\n(ticket / hand-off)
Refusal response\n(policy-based)
Observability
Trace & policy logs
Token / cost meter

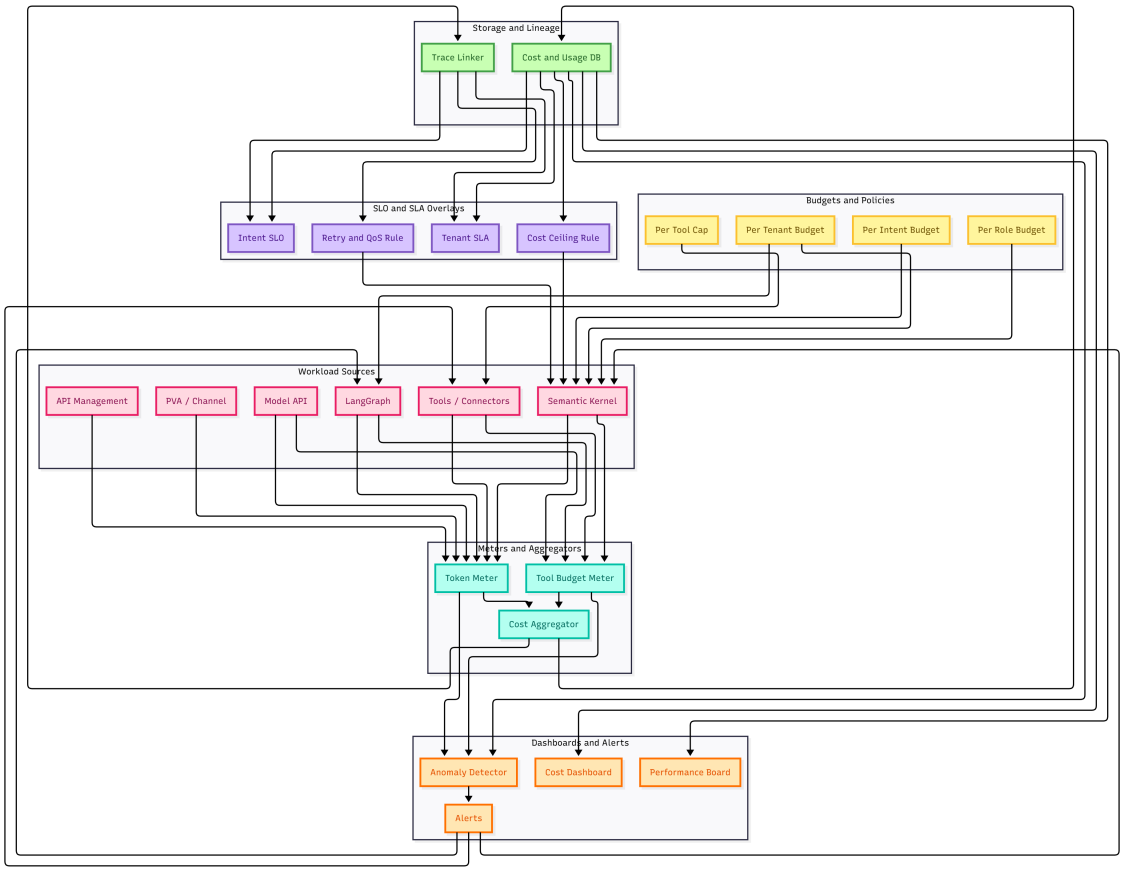



Figure 25: Pattern 25: Cost Monitoring Topology — Token Meters, Tool Budget Caps, and Per-Intent SLO/SLA Overlays

### 7.4 Evaluation Pipeline and GRC Dashboard

**Evaluation Pipeline.** Nightly automated evaluation using Promptflow or SK flows, with metrics like:

- Factuality.
- Citation coverage.
- Refusal accuracy.
- Safety incidents.
- Latency.

**GRC Command Dashboard.** A Power BI (or equivalent) view for executives and GRC leaders, showing:

- **Governance**: policy adherence and changes.

- **Risk**: trends in incidents and mitigations.
- **Compliance**: control coverage and audit status.
- **Assurance**: quality metrics, eval scores, SLO/SLA performance.

This closes the loop from operational telemetry back to leadership visibility and decision-making.

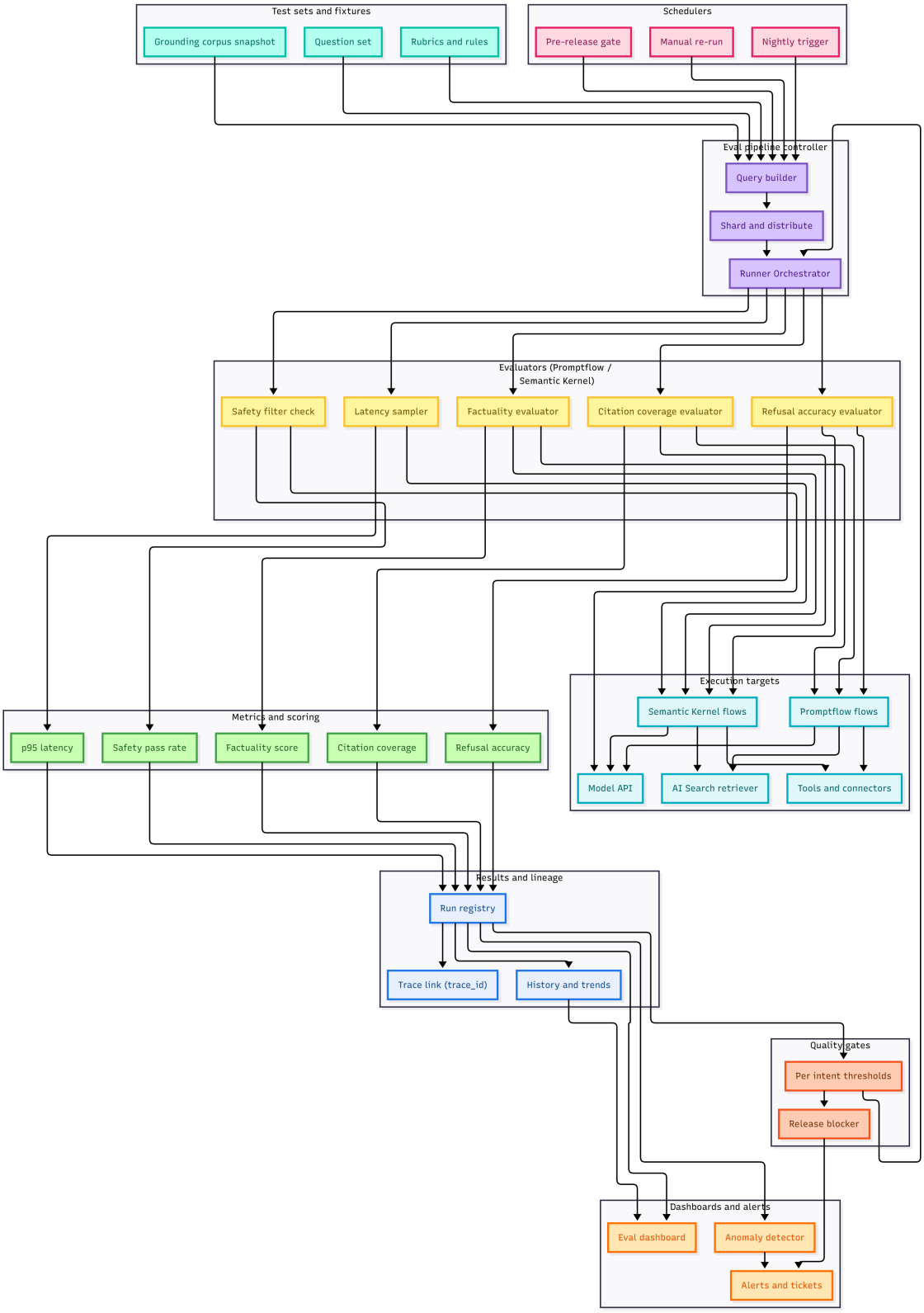



Figure 26: Pattern 26: Evaluation Pipeline — Automated Promptflow & Semantic Kernel Jobs for Model Quality Assurance

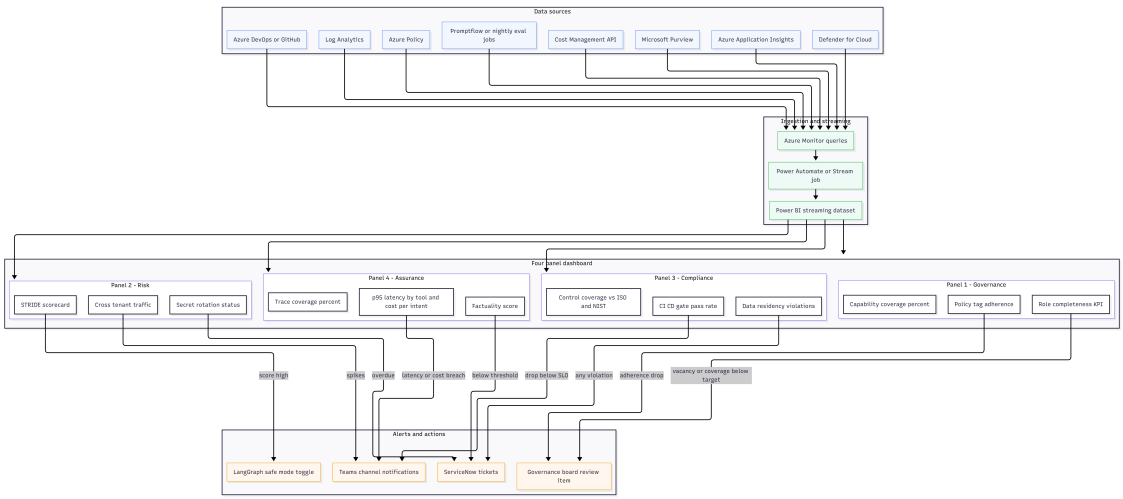



Figure 27: Pattern 30: GRC Command Dashboard

# 8. Operations and Lifecycle Management

## 8.1 Deployment and CI/CD

The architecture described here includes explicit patterns for deployment and continuous delivery.

**Deployment Diagram.** Separate **Prod** and **Non-Prod** environments with:

- App Service / Container Apps, Functions.
- Storage tiers (hot/cool/archive).
- AI Search tiers aligned to cost and performance requirements (e.g., Basic vs Standard Vector).

**CI/CD Flow.** Versioning and blue/green deployments for:

- Orchestrator code.
- LangGraph workflows.
- Semantic Kernel tool bundles.
- Prompt templates and flows.

Quality gates ensure that new releases are safe, reversible, and auditable.

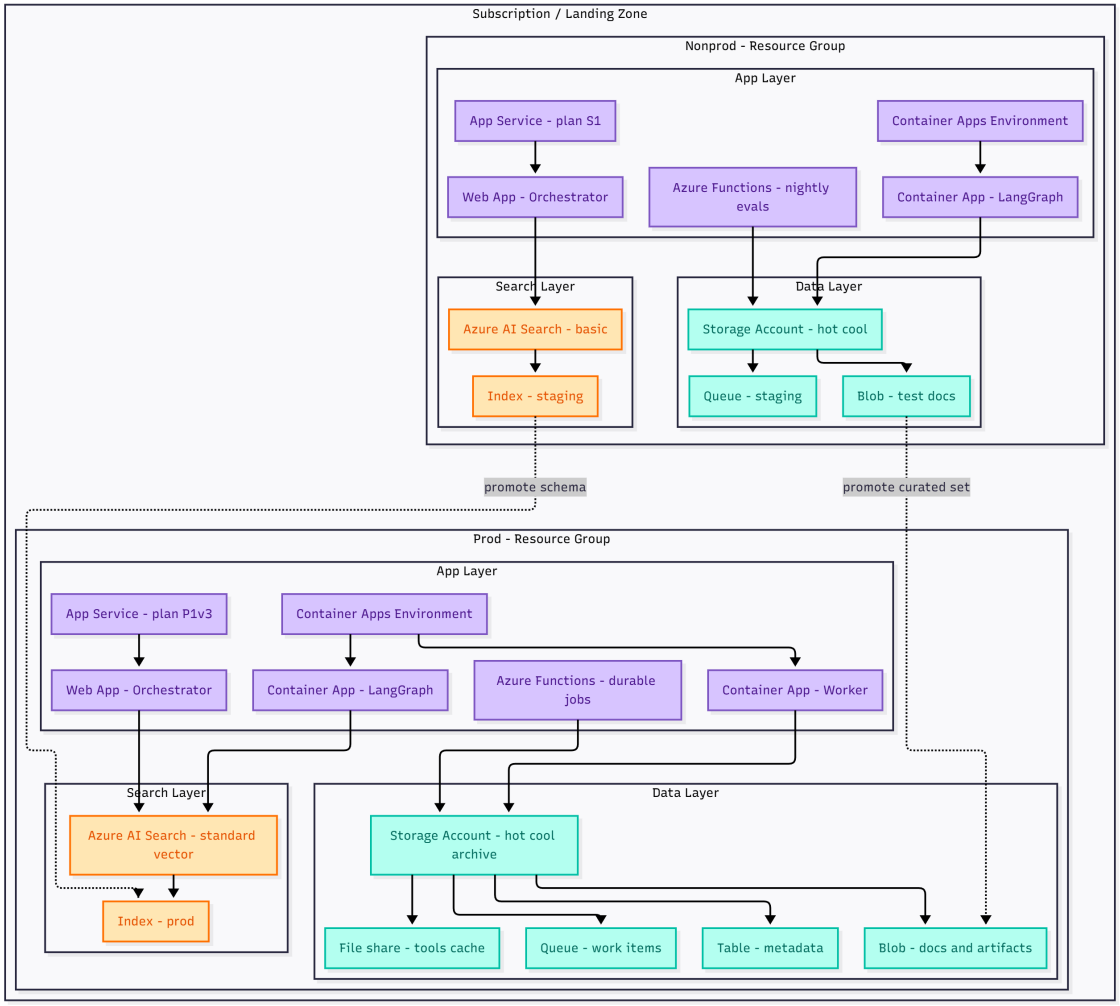



Figure 28: Pattern 27: Deployment Diagram — App Service/Container Apps, Functions, Storage Classes, and AI Search Tiers (Prod & Non-Prod)

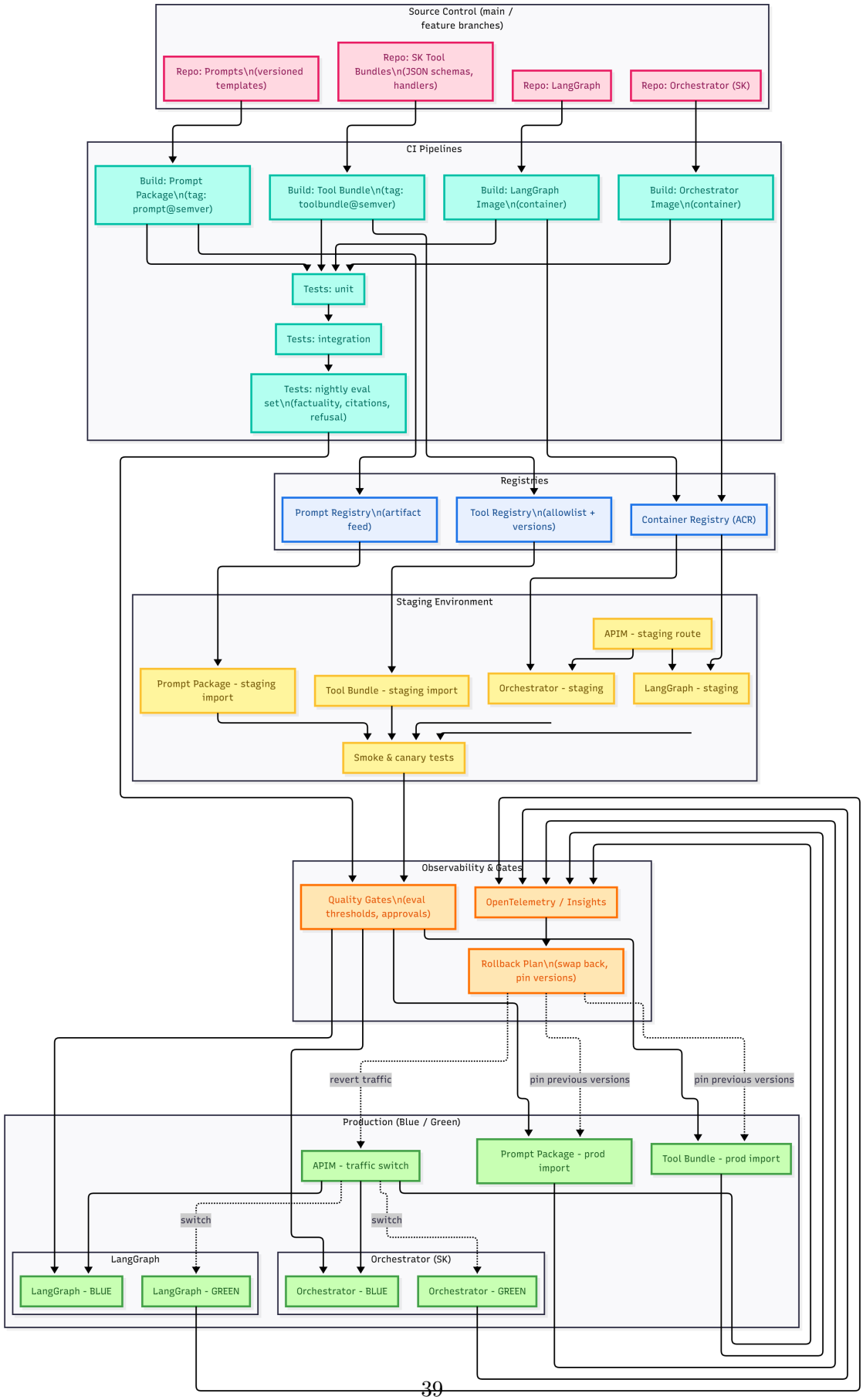





Figure 29: Pattern 28: CI/CD Flow — Prompt Versioning, Semantic Kernel Tool Bundles, and Blue/Green Deployments for Orchestrator & LangGraph

### 8.2 Runbook Flows

The **Runbook Flow** defines standardized, auditable procedures for:

- Hotfix deployments.
- Traffic rollback.
- Index rebuilds.
- Secret rotation.

Each runbook ties into telemetry and GRC requirements, minimizing mean time to recovery (MTTR) while preserving compliance and trust.

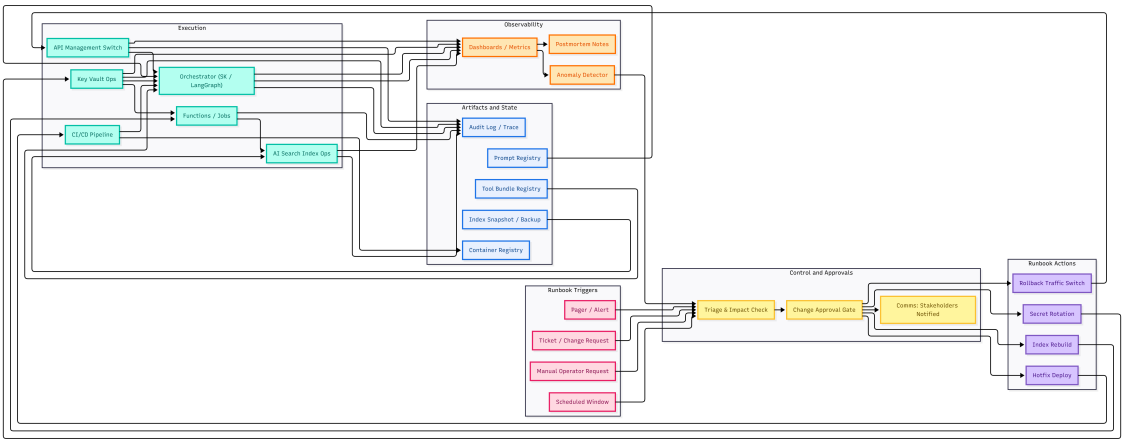



Figure 30: Pattern 29: Runbook Flow — Hotfix Deployment, Traffic Rollback, Index Rebuild, and Secret Rotation

## 9. Conclusion

The architecture described in this whitepaper defines a comprehensive control plane for agentic AI:

- The **Capability Map** (Assist, Orchestrate, Retrieve, Govern) provides a stable organizing principle.
- **Stakeholder maps and value stream views** align people, process, and technology.
- **C4 system, container, and trust-boundary diagrams** ensure a robust, secure runtime environment.
- **BPMN, state machine, and sequence diagrams** make runtime behavior explicit and testable.
- **Data flows, logical models, and tool registry patterns** deliver high-quality, grounded knowledge access.

- **GRC overlays, telemetry, eval pipelines, and dashboards** embed governance and assurance into daily operations.

Taken together, these patterns form a blueprint for designing responsible, explainable, and governable agentic AI systems on Azure—systems that can scale across use cases and business units while remaining controllable, compliant, and aligned with organizational values.