



Enterprise Architecture for Agentic AI

**Microsoft's approach to scaling agent adoption
while staying in control of security and governance**



4

Introduction

5

Central Hub
definition

7

Enabling
multi-runtime
governance

7

Intelligent Data
Platform

8

Foundational
capabilities for data
products

9

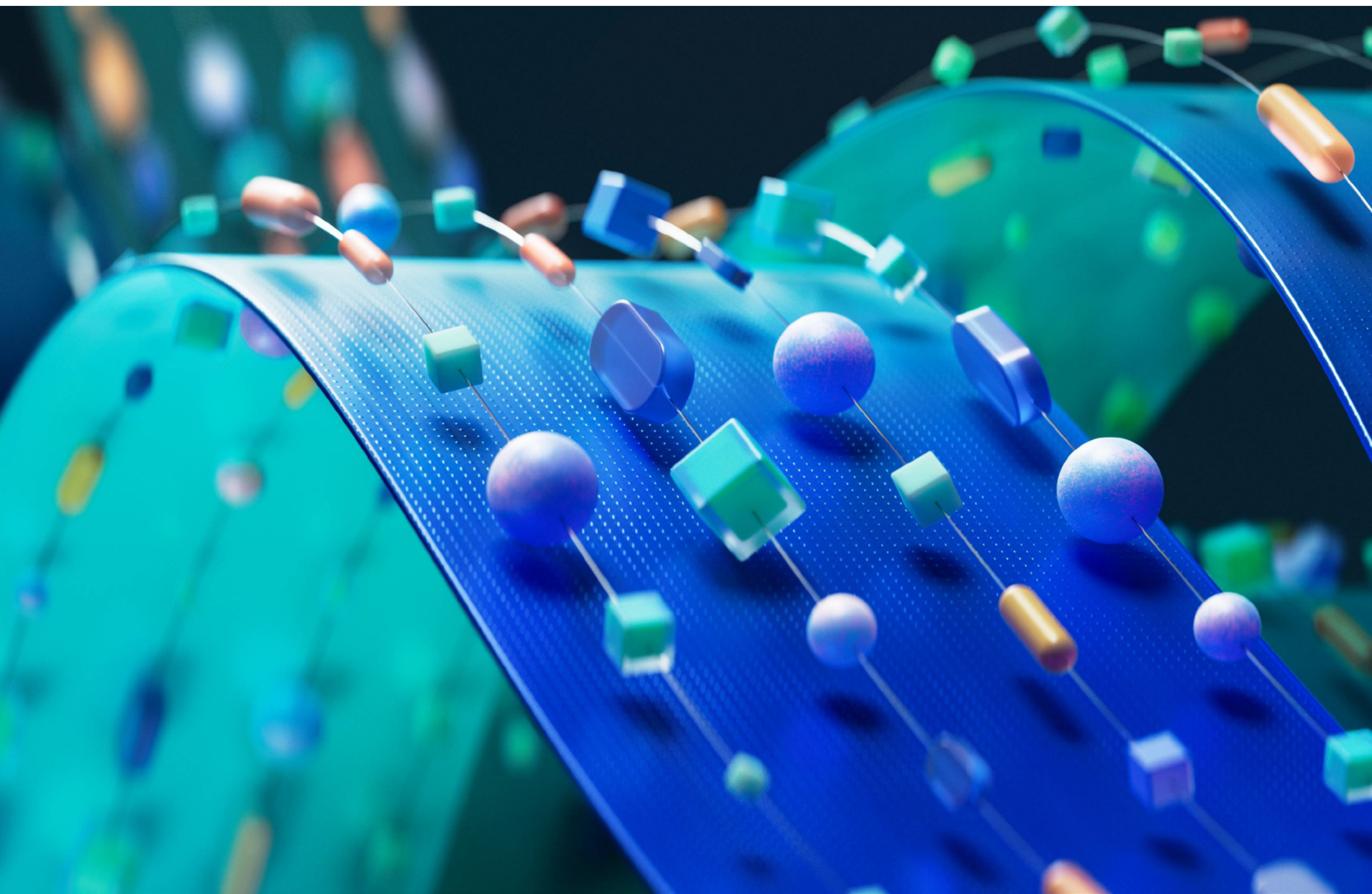
Application Plane

11

Landing zone
patterns and
responsibilities

12

Microsoft's workshop
and incremental
delivery for agent mesh



Executive Summary

Scaling agents across the organization with reliability, security and efficiency in mind is very difficult to do. The recommended solution is to implement an enterprise agent mesh as a means to implement a concept for application plane for business-data-aware enterprise agent platform using a three-plane, hub-and-spoke architecture: (1) a central hub that provides the single governed path to models and tools (identity, policy enforcement, inventory/registry, audit, and telemetry), (2) An intelligent data platform accessible via standard protocols (e.g., MCP), delivering augmented-data-products, that is, data assets augmented with the operational and analytic models that define their semantics, and (3) an application plane of standardized per-workload landing zones that host agent runtimes and integrations while consuming hub-mediated endpoints. First, adopt this architecture by deploying the hub's gateways and inventory, as well as an enforceable onboarding blueprint. Second, onboard a small set of domain MCP servers and a few representative agent workloads. Finally, scale using templates, lifecycle automation, and policy-as-code. The recommended solution builds and operates an Agent Marketplace for the frontier company using Microsoft Fabric, Microsoft Foundry, and Copilot Studio as your core agent platforms while allowing provisions in-place to expand to others.



Introduction

Agentic AI is no longer a research topic. It is becoming an operating layer of the enterprise. Across every industry, organizations are moving from single-purpose AI features to populations of agents—autonomous software entities that reason, retrieve, act, and coordinate on behalf of users and business processes. This shift is comparable in magnitude to the arrival of the public internet: value will not come from any one model, framework, or vendor, but from the connective tissue that lets agents discover one another, share context, invoke enterprise resources safely, and operate under verifiable governance.

That connective tissue is now coalescing into an emerging set of open standards. Model Context Protocol (MCP) is becoming the universal way for agents to invoke tools and access knowledge. The Agent-to-Agent (A2A) protocol is establishing how agents discover each other and delegate work across organizational boundaries. Agent identity standards built on DNS, public-key infrastructure, OIDC, and JWT are giving every agent a verifiable principal. OpenTelemetry's GenAI semantic conventions are providing a common observability fabric so every reasoning step, tool call, and decision is traceable.

Enterprises are moving from isolated AI proofs-of-concept to portfolios of agents operating across business units, geographies, and technology stacks. The core challenge is scaling many agents safely and predictably. An agent mesh addresses this by providing an enterprise-wide pattern for how agents discover and invoke models, tools, business data, and other agents - so teams can keep execution decentralized (choosing the runtime that fits each workload) while using a consistent, governed interaction model.

In practice, the mesh centralizes the cross-cutting controls that otherwise get rebuilt (and inconsistently implemented) in every project: it establishes governed gateways and registries as the supported path to models and MCP-exposed capabilities, standardizes identity propagation and authorization policy, and requires shared audit trails and operational telemetry so actions can be traced end-to-end across runtimes. By making these controls template-driven and reusable, the enterprise reduces duplicated platform work, prevents governance bypass via direct backend connectivity, improves observability and incident response, and can answer basic control questions at scale - who

accessed what, which tools executed, under which identity, and at what cost.

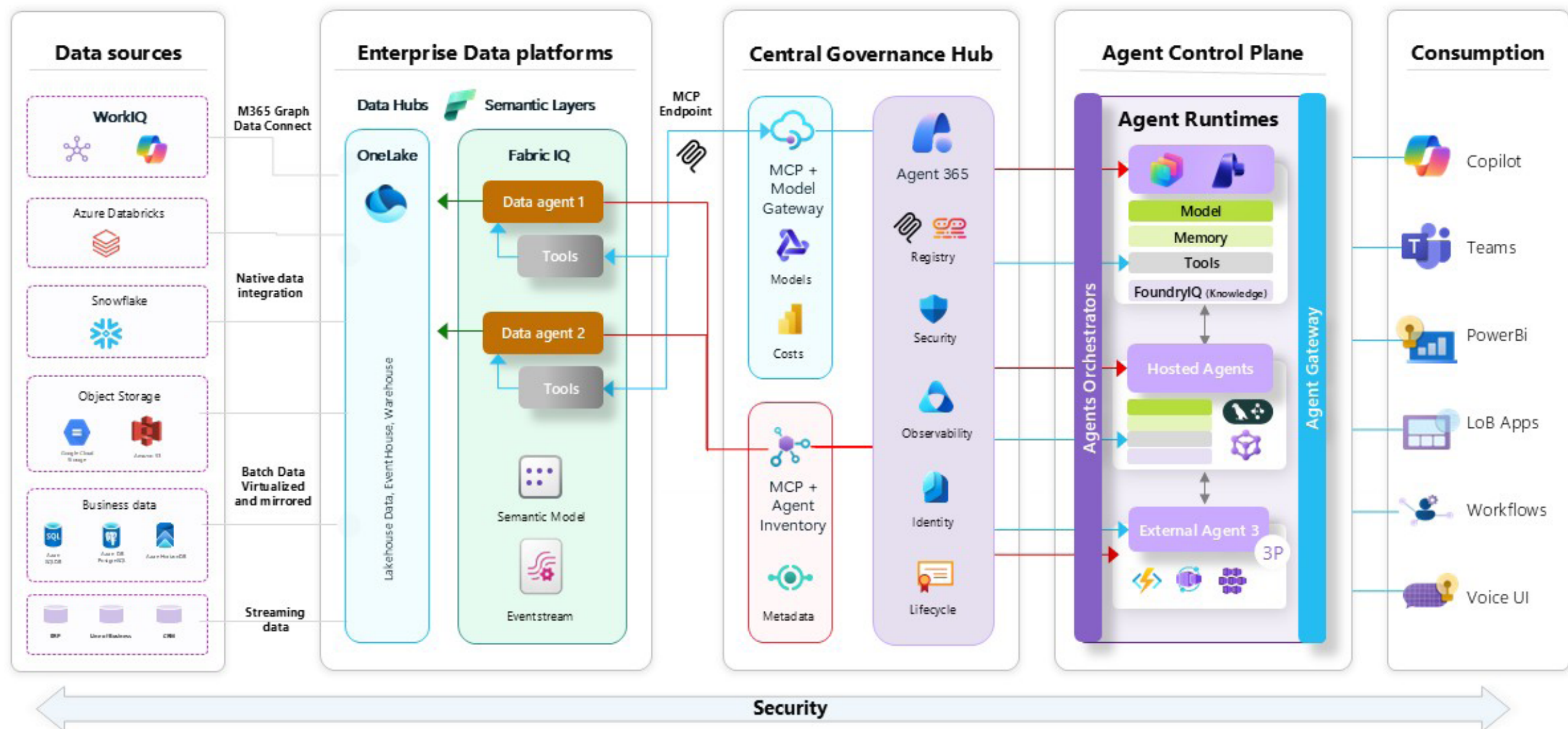
This paper defines the application plane of an agent mesh powered by contextualized business data.

For agents to retrieve the appropriate business data context, two conditions must be satisfied. First, data must be trusted. Throughout this document, we therefore assume a Data Mesh architecture, as it natively provides the security and governance capabilities required to make data products trustworthy and to reduce the risk of AI driven data leakage. Second, data must be understood. To this end, data products are enriched with business knowledge representations (such as ontologies) and supported by graph data engines that capture and exploit relationships across heterogeneous operational datasets - an approach increasingly realized through Context Graphs. Accordingly, in this document data products denote the combination of operational and analytical data assets, governance and security policies, and the abstractions and engines that enable agents to retrieve contextualized business data.

In this model, the goal is to reach autonomy. Agents built by application teams can securely and autonomously connect to data products, tools, and other agents through central governance. Domain teams publish governed data products. Application teams consume those products and compose them into agent solutions, delivered through user interfaces, workflows, and system integrations. The application plane defines the runtime landing zones, integration patterns, and operational controls required to deploy and run these solutions in production.

The paper recommends a three-plane architecture organized in a hub-and-spoke topology. A central hub provides governance and mediation services that standardize how models, tools, and agents are exposed and consumed. A data plane enables domain teams to publish data products which ultimately allow agents to reason over enterprise data within a shared business understanding. An application plane provides per-workload landing zones for hosting agents across runtimes, orchestrating multi-agent solutions, and integrating into channels and enterprise systems.

Agentic setup – logical architecture



Assumptions for the scope of this paper: private network connectivity is the target state for models, gateways, agents, and MCP servers; identity and authorization use Microsoft Entra ID; the enterprise operates a private network where shared platform services run in the hub subscription and workload-specific agent runtimes run in spoke subscriptions; data products are defined as above, and are delivered in Microsoft Fabric.

Key recommendation: design and implement the application plane as a platform capability with enforceable templates for security, networking, observability, lifecycle automation, and integration contracts. Do not allow project teams to embed direct model or tool endpoints. Require consumption through hub-mediated interfaces so the enterprise can scale agent adoption without losing compliance, accountability, or operational reliability.

Central Hub

The enterprise control surface for models, tools, and agent interactions.

The central hub is the enterprise control surface for models, tools, and agent interactions. It provides the recommended path for agents and applications to reach model endpoints,

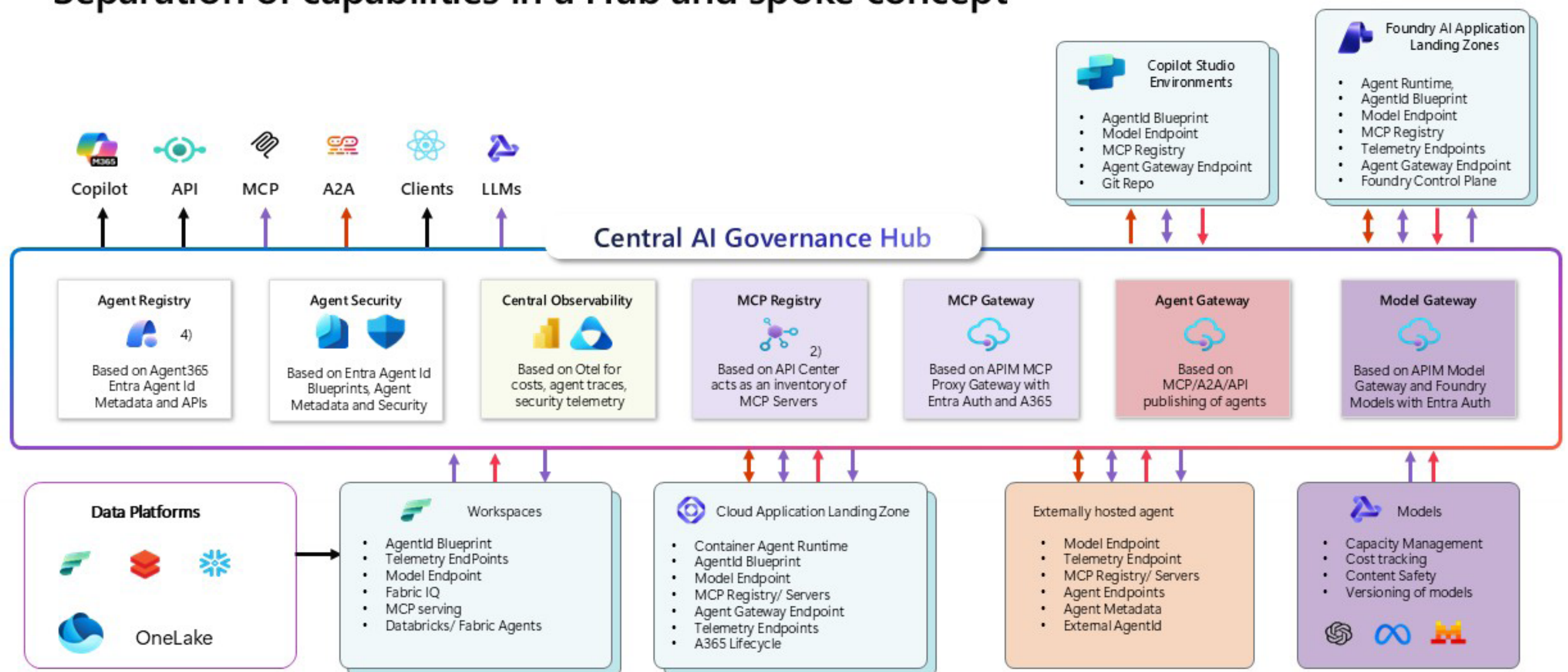
MCP servers, and enterprise APIs, and it enforces identity, policy, auditability, and telemetry consistently across all spokes.

The hub consists of 2 components:

- API Management (model gateway) as policy enforcement and routing layer
- MCP Inventory as system of record for tools and data

The first hub component is the model gateway: the enterprise entry point for consuming language models and related AI services without exposing raw endpoints or distributing credentials. Implemented typically with Azure API Management as the policy enforcement and routing layer, it centralizes capacity management (routing, rate limits, quotas, and regional controls), enables transparent cost attribution per consuming workload, and provides a stable contract that decouples applications from underlying providers and deployments. By owning model versioning and lifecycle responsibilities - promotion, deprecation, and retirement - within the gateway, the platform team can evolve model deployments safely while application teams integrate once and receive predictable throughput and governed access.

Separation of capabilities in a Hub and spoke concept



The second hub component is an MCP inventory that acts as the system of record for tools and data interaction surfaces exposed through Model Context Protocol, paired with an MCP gateway that surfaces those MCP servers through Azure API Management. MCP standardises tool discovery and invocation, but the enterprise still needs inventory discipline to capture ownership, purpose, classification, lifecycle state, required authentication, and approved consumers. Microsoft Azure API Center is an example of an inventory capability for storing this API and tool metadata, while API Management provides the mediated interaction surface so consumers discover and call MCP capabilities through stable, governed endpoints rather than coupling directly to backend MCP server addresses.

For MCP servers, two non-negotiables apply: they must be registered in the inventory and they must be consumed through the gateway. Registration requires complete metadata (named owner and operational contact, classification/compliance tags, lifecycle state/versioning intent, and declared consumer scope). Gateway onboarding requires Microsoft Entra ID-based authentication with no shared secrets, automated policy validation (token and claim

checks, access control rules, and any required safety/data-handling constraints), and mandatory observability and audit fields so every tool invocation can be traced and attributed consistently. MCP servers that cannot meet these requirements are treated as out of scope until they can be remediated.

A critical property of the central hub is that it separates what must be centralised from what can remain federated. Identity policy, baseline enforcement, inventory, and central telemetry must be centrally owned because inconsistency in these areas becomes systemic risk. Tool implementations, data products, and agent logic should remain federated because that is where domain knowledge and delivery speed matter most. The hub therefore functions as an enablement and enforcement platform rather than as a central development environment. It does not compete with domain teams; it creates a consistent substrate on which domain and application teams can build.

Lifecycle automation, audit traceability, and how the hub enables multi-runtime governance

The hub's technical components only translate into enterprise value when governance is operationalized as lifecycle discipline. In an agent mesh, governance is not a one-time review; it is a continuous property that must hold as agents evolve, as tools change, and as data products are versioned. The hub therefore needs a lifecycle automation model that defines how artifacts are published, validated, promoted across environments, and retired.

A practical lifecycle model begins with a publish-and-consume contract: a declarative artifact stored in source control that registers a model connection, MCP server endpoint, or agent endpoint into the enterprise inventory and gateway configuration, while also defining the stable interface that consumers integrate against. The contract pairs a gateway URL and authentication pattern with required metadata such as ownership, data classification, intended consumers, and operational contacts, so application teams can depend on a consistent endpoint and interface definition without needing to know backend addresses.

Audit traceability and policy-as-code are the hub's non-negotiable controls: the enterprise must be able to reconstruct what an agent did, which tools and data products it invoked, under which identity, and under which policy decisions, with telemetry that can be correlated end-to-end across runtimes. To make that traceability enforceable rather than aspirational, policies are centrally defined, versioned, and applied consistently at the gateway layer (for example via Azure API Management policies for token validation, claim checks, routing constraints, and rate limits) and across the rest of the platform through Azure Policy on the underlying Azure resources. Together, gateway enforcement plus resource-level policy baselines ensure governed access paths, consistent authentication and authorization semantics, and reliable evidence for audit, incident response, and cost and risk attribution. The hub becomes even more important when an enterprise aims to connect agent governance to broader organizational governance constructs. A registry-centric model, where agents are inventoried, governed, and managed with

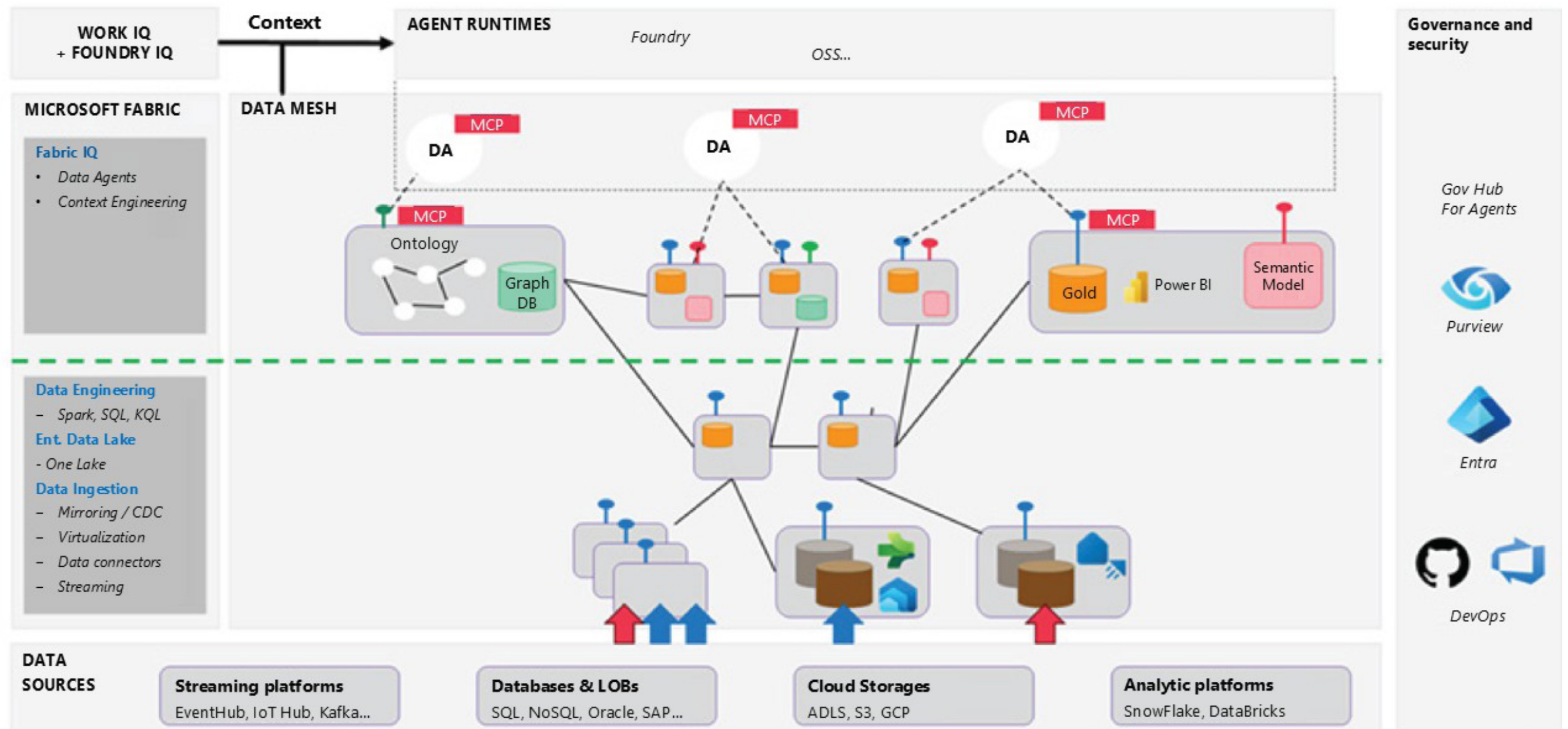
lifecycle controls, aligns with the direction of Microsoft Agent 365 as a control plane concept for agents. Even if an enterprise adopts Microsoft Agent 365 gradually, designing the hub inventory and lifecycle semantics in a way that can align with agent registry and access control patterns reduces future integration friction. It also reinforces the enterprise expectation that an "unregistered agent" is an unauthorized agent, and that every agent must have clear ownership, permissions, and lifecycle status.

Intelligent Data Platform

The data plane provides the foundational capabilities for domain-owned data products to be published and consumed with predictable semantics. For agents to reason consistently about business data, that data must be both trusted and clearly understood. A data mesh provides the foundation for trust by enforcing governance, ownership, and security at the level of individual data products. On top of this trusted foundation, knowledge models are essential to help agents interpret both analytical and operational data, allowing them to assemble the right business context rather than isolated facts. This enables agents to reason holistically across domain specific data coming from multiple, independently managed data products. To operate at scale, these data agents and knowledge models should be exposed through standard access protocols (such as MCP) and centrally governed through a governance hub, ensuring consistency, control, and traceability across the organization

In this context, Microsoft Fabric forms the foundation of an intelligent data platform for agentic decision making. It brings together several core components: an enterprise data lake, data governance & security, knowledge models, data agents and access interfaces.

Intelligent Data Platform for Agents



Foundational capabilities for domain-owned data products that can be consumed by agents and applications

Microsoft Fabric's OneLake provides an enterprise data lake that unifies operational data across multiple sources, enabling analytics and providing a consistent, scalable foundation for agentic applications. Fabric offers multiple native ingestion patterns: database mirroring continuously replicates data from on premises and cloud databases, shortcuts enable virtual access to external platforms such as Databricks, Snowflake, and major cloud object stores without data movement, and batch or streaming ingestion is supported through Fabric Pipelines and Real Time Intelligence. All data in OneLake is stored in a common format (Apache Delta) and can be queried with SQL or Spark, with secure access enforced through Microsoft Entra ID and fine grained permissions down to row and column level.

As agents scale across an organization and they act autonomously on business information, data governance

becomes the foundation of trust, guaranteeing that agent decisions are based on accurate and authorized data. This aligns naturally with Data Mesh principles, where data is organized as domain owned products with clear ownership, quality guarantees, explicit and observable governance and access policies. Microsoft Fabric enables this model by supporting governed data products within a unified platform, while Microsoft Purview provides comprehensive governance capabilities across the data lifecycle. Together, Fabric and Purview deliver data discovery and cataloging, automated classification and sensitivity labeling, end to end lineage tracking, policy based access controls, and auditing of data access and usage. These capabilities ensure that agents consume only authorized data, that inputs to agent decisions are traceable and inspect-able, and that organizations can demonstrate compliance, explain outcomes, and apply safety controls consistently across agentic workflows.

Knowledge models enable transforming business data into task specific contexts - one of the hardest challenges for agents.

Without an explicit understanding of business meaning, agents often misinterpret relationships across datasets, leading to incorrect inferences and uncontrolled hallucinations that fundamentally limit scalable, autonomous deployments. Because data tables do not encode business meaning, they are often embedded directly into prompts: while workable for isolated cases, this approach is fragile, hard to test, and costly to maintain across agents and scenarios. Fabric IQ addresses this gap by treating knowledge models as shared, governed assets: by explicitly representing business entities, relationships, and metrics in a machine readable form, knowledge models ground agent reasoning in verifiable structures rather than inferred assumptions. This grounding reduces hallucinations and enables explainability: agent decisions can be traced to well defined business concepts, making behavior inspectable, auditable, and trustworthy at scale.

Fabric IQ provides two complementary knowledge model types:

- Ontologies represent operational models by explicitly defining business entities and the relationships that connect them. Built on graph technologies, they allow agents to reason over connected data by traversing complex relationships. These graph based structures form Context Graphs, enabling operational intelligence such as identifying customers in a specific region who are associated with recent purchases and unresolved product support issues.
- Semantic models represent analytical knowledge by explicitly defining metrics and dimensions and the rules that govern their aggregation, calculation, and interpretation. Grounded in established business intelligence concepts, they provide agents with an explicit, machine readable representation of business analytics. By leveraging optimized columnar storage, semantic models efficiently support aggregation and analysis patterns driven by business dimensions.

Patterns for publishing via MCP, governing read versus mutation, and connecting semantics to retrieval

Microsoft Fabric Data Agents are purpose built and systematically tested to retrieve, correlate, and reason over enterprise data through multi step, multi source data conversations. They translate natural language requests into structured reasoning plans, orchestrating controlled data retrieval across sources such as Fabric IQ knowledge models, Delta tables in OneLake, and Azure AI Search indexes. By combining retrieved facts with explicit business semantics, Data Agents assemble grounded context that supports reliable reasoning and consistent, task relevant outcomes. The exposure layer is what makes data products reliably consumable by both agents and applications. By providing stable interaction surfaces - such as query endpoints, data agents, and agent compatible protocols - it ensures predictable semantics independent of the underlying platform or runtime, allowing agents to reason coherently across analytical and operational data from multiple domains. At scale, this exposure layer must be built on standard access protocols like MCP and governed through a centralized governance hub. MCP enables discoverable and reusable access to data products and knowledge models, giving agents visibility into both the available data and its semantic and governance context. Central governance enforces consistent access control, policy, lineage, and lifecycle management across heterogeneous environments, while treating MCP servers as first class operational products introduces discipline around versioning, observability, and deprecation. Together, these capabilities extend Data Mesh principles into the application plane, enabling enterprise wide reuse of agent capabilities with consistency, control, and traceability - without degenerating into ad hoc integrations.

Application plane

The application plane is the enterprise capability that turns governed assets into deployed solutions. It provides the runtime environments in which agents execute, the integration mechanisms that connect agents to tools and data products, and the operational controls required for production reliability.

In a hub-and-spoke enterprise design, each application landing zone is a spoke subscription boundary that hosts per-workload resources under the ownership of an AI project team. The project team is responsible for solution delivery and operations within its boundary, while the central platform team owns shared governance services such as the model gateway, the MCP gateway, and enterprise inventories. This separation matches how enterprises staff and operate, because the central team enforces baseline controls while project teams build and run workloads.

A robust application landing zone must provide a standardised foundation for identity, networking, governance, monitoring, and cost management. It should include consistent use of managed identities, clear role-based access control boundaries, private network integration with the hub and data plane, and standardised telemetry pipelines. The goal is that application teams can deploy production-ready agent solutions without violating corporate policies and without reinventing foundational infrastructure for every project.

The application plane must support multiple agent hosting formats because enterprise scenarios require different trade-offs between convenience, control, extensibility, and operational responsibility. These formats are:

- No-code / low-code agents
- Hosted agents tied to Microsoft Foundry
- Custom container-hosted agents (outside of Microsoft Foundry)

The first category is no-code and low-code agents implemented as prompt agents and workflows within Microsoft Foundry Agent Service, tied to a Microsoft Foundry project. In this category, teams use managed capabilities for orchestration, evaluation, and telemetry within the project boundary. They leverage tools exposed through MCP, grounding through connected knowledge sources, and project-scoped observability for tracing and evaluation.

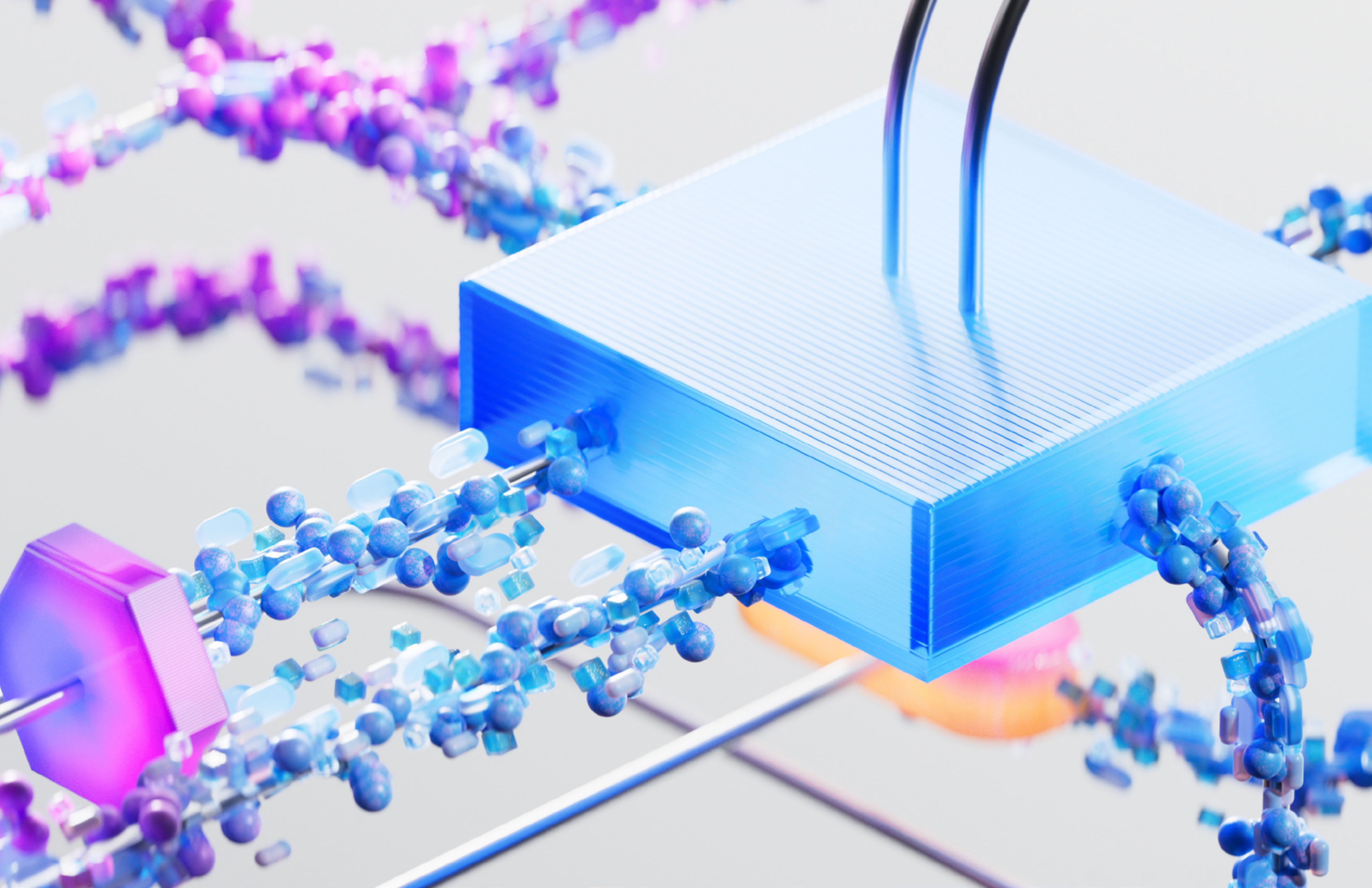
The second category is hosted agents tied to a Microsoft Foundry project but executed as container workloads built from code. In this category, teams develop agents with full software engineering practices, package them into containers, and deploy them into controlled runtimes while still integrating with project-level orchestration and governance. This pattern is useful when teams need custom

dependencies, custom tool integration logic, specialized orchestration behaviors, or specific runtime control that is not available in purely no-code environments. It also allows mixed composition, where low-code workflows delegate specific tasks to pro-code hosted agents, enabling a pragmatic combination of rapid assembly and deep customization.

The third category is custom container-hosted agents outside of Microsoft Azure AI Foundry, including agents running on Azure Container Apps, Azure Kubernetes Service, or potentially in non-Azure environments. In this category, the enterprise still requires integration with hub capabilities through standard protocols such as REST, MCP, and agent-to-agent communication patterns leveraging Foundry APIs and capabilities. The architectural objective is that these agents remain governed by consuming hub-mediated models and tools, using identity patterns that avoid secret distribution, and emitting telemetry that can be correlated with the broader mesh observability model.

A crucial point for the application plane is that model and tool access should be mediated by the central hub, using template-driven patterns rather than embedded endpoints and credentials in each agent. This keeps policy enforcement and auditability consistent, lets platform teams manage capacity, safety configuration, and cost controls centrally, and enables application teams to onboard quickly by instantiating approved templates and reference implementations (for hosting, integration to hub-mediated endpoints and MCP-exposed data products, and telemetry) instead of reinventing infrastructure for every project.





Cross-project agent integration patterns and operational responsibilities of the landing zone

The defining capability of an application plane in an agent mesh is that it enables interaction across projects, runtimes, and organizational boundaries without devolving into point-to-point coupling. A mature application plane provides integration patterns that support four common scenario families while preserving governance semantics.

The scenarios are:

- Data-centric enrichment and indexing workflow
- Interactive user-serving assistants
- Sequential and group workflow
- Agent-to-agent collaboration and dynamic planning

The first scenario family is data-centric enrichment and indexing workflows. In these scenarios, agents perform ingestion, parsing, chunking, classification, and enrichment

over structured and unstructured content. The output is typically a governed retrieval surface such as a search index, an enriched dataset, or a curated knowledge store. These workflows often require batch processing, integration with data plane products, and careful handling of identity and auditability because they may touch sensitive content. The application plane should provide patterns for running these workflows as durable processes, with explicit scheduling, retries, and telemetry, rather than as ad-hoc interactive chats.

The second scenario family is interactive user-serving assistants integrated into channels such as Microsoft Teams or web applications. In these scenarios, latency, identity propagation, and user experience are primary concerns. The application plane should provide standard patterns for connecting channel identity to agent identity, preserving on-behalf-of context where required, and ensuring that tool calls and data product access respect user permissions. This pattern benefits from combining Microsoft Copilot Studio as a channel and orchestration layer with specialized agents implemented in Microsoft Foundry Agent Service or container runtimes when deeper control is required.

The third scenario family is sequential and group workflows that implement business processes with one or more agents. These workflows may be long running, may require human approvals, and may integrate with enterprise systems such as customer relationship management, enterprise resource planning, or IT service management platforms. The application plane should provide orchestration patterns that externalize workflow state and logic rather than hiding it in prompts. It should support human-in-the-loop steps as explicit workflow states, and it should support escalation paths for compliance and accountability.

The fourth scenario family is agent-to-agent collaboration and dynamic planning. In these scenarios, an agent decomposes a goal into tasks, discovers suitable specialized agents, delegates tasks, and aggregates results. This pattern only becomes safe at enterprise scale when agents discover and invoke other agents through governed registries and gateways rather than through informal endpoint sharing. The application plane should therefore integrate with a central agent gateway or registry concept and should preserve identity and policy semantics across agent-to-agent calls. In a mature mesh, this includes dynamic handoff, capability-based routing, and explicit constraints that prevent an agent from delegating tasks to unauthorized or untrusted agents. The enterprise value is that specialist agents become reusable building blocks and that complex solutions become compositions of domain capabilities rather than monolithic systems.

The landing zone must provide a standardized environment for hosting and operating AI-native applications built on Microsoft Foundry, Azure Container Apps, and Azure Kubernetes Service, ensuring that identity, networking, governance, monitoring, and cost management concerns are implemented consistently. It should treat Microsoft Azure AI Foundry as a control plane for model hosting, prompt orchestration, and agent runtime integration, while treating container runtimes as isolation boundaries for deploying orchestration logic, MCP-compatible agent services, and application middleware.

The landing zone accelerator template provides a deliberate private networking posture, using virtual networks, private endpoints, network security groups, and private DNS resolution patterns to eliminate public ingress paths where feasible between agents, gateways, telemetry services, and managed retrieval stores. Managed identities should be the default mechanism for service-to-service access to avoid

credential distribution. The landing zone brings defaults OpenTelemetry-style signals and standardized telemetry pipelines so that traces and metrics can be correlated across the mesh.

Finally, the landing zone accelerator enables deployment automation through infrastructure-as-code templates and continuous delivery pipelines. It automates resource tagging, policy assignment, monitoring configuration, and baseline security controls so that compliance remains consistent and auditable. By implementing these responsibilities as templates, application teams can innovate quickly within their spokes while staying within enterprise guardrails.

Integration concept: how to build the agent mesh through a workshop and incremental delivery with Microsoft

An enterprise agent mesh is not implemented successfully through a single architecture document or a central platform build-out in isolation. It is implemented through a decision-driven workshop and a controlled series of increments that validate assumptions through real workloads. The integration concept described here connects the central hub, the data plane, and the application plane into a practical delivery model that an enterprise can execute jointly with Microsoft delivery teams.

The first step is to plan the central hub as the foundation, because the hub defines the non-negotiable governance decisions that shape all spokes. The enterprise should establish the model gateway pattern, the MCP inventory model, and the gateway mediation surfaces early, even if they begin with a minimal viable scope. The goal is not to build a perfect hub from day one; the goal is to create a supported path that makes governed consumption easier than bypassing governance. The hub must also define the default blueprint that encodes baseline security policies and audit traceability requirements. When the blueprint exists as deployable templates, application and data teams can onboard with predictable outcomes and without repeated design debates lifecycle and governance, not ad-hoc datasets.



The third step is to onboard a small set of application teams and implement representative solution scenarios that stress the mesh. These scenarios should include at least one user-serving interactive assistant integrated into a channel such as Microsoft Teams, at least one data-centric enrichment pipeline that consumes data plane products, and at least one workflow-oriented business process scenario that includes tool invocation and optional human-in-the-loop steps. These implementations should be deliberately chosen to force decisions about hosting patterns, identity flows, tool governance, and observability correlation.

The foundational concepts that enable scaling are agent identity, policy baselines, observability semantics, and templating discipline. Agent identity ensures that every agent has clear ownership, permissions, and lifecycle status, and that actions can be attributed even in autonomous or semi-autonomous execution. Policy baselines ensure that authentication, authorization, safety controls, and data access constraints are consistent across runtimes. Observability semantics ensure that the enterprise can trace interactions across agents, tools, and data products,

correlate failures, and attribute cost and risk. Templating ensures that these concepts become deployable reality rather than aspirational guidelines, allowing teams to scale adoption through repeatable instantiation.

The final result of this paper is a coherent, implementable model for building an enterprise platform in which agents collaborate and communicate through governed contracts, data products are exposed as reusable MCP capabilities, and application teams deliver intelligent solutions across channels and systems without breaking enterprise controls. The implementation should happen incrementally. The enterprise should begin with a minimal hub and a small number of published products and agents, then expand the inventory, strengthen policies, improve telemetry correlation, and scale templates as adoption grows. The architecture should be treated as a living system that evolves with product capabilities, enterprise learnings, and regulatory constraints. A center of excellence team should own the living architecture, maintain the templates and governance model, and continuously improve the onboarding experience, while domain and application teams retain autonomy to innovate within their spokes.

Use the Microsoft Agent Factory to establish your own enterprise architecture for agents

Microsoft Agent Factory is a unified program that enables organizations to build, deploy, and scale AI agents across the full lifecycle using tools like Copilot Studio, Microsoft Foundry, GitHub Copilot, and Microsoft Fabric. It simplifies adoption through a single metered commercial model (P3), which reduces licensing complexity and provides predictable, flexible spending.

The platform integrates intelligence layers—Work IQ, Fabric IQ, and Foundry IQ—to ground agents in organizational context, enterprise data, and advanced reasoning, improving accuracy and relevance. It supports a role-based development model, enabling both business users and developers to collaborate using low-code and pro-code tools.

With access to Forward Deployed Engineers and comprehensive skilling programs, organizations can accelerate from experimentation to production. Built-in governance and security ensure agents operate reliably at scale, helping organizations realize measurable business outcomes.

See Microsoft Agent Factory →

Source:

1 <https://learn.microsoft.com/azure/cloud-adoption-framework/ai/>

2 <https://learn.microsoft.com/azure/cloud-adoption-framework/ai-agents/>

©2026 Microsoft Corporation. All rights reserved. This document is provided "as-is." Information and views expressed in this document, including URL and other Internet website references, may change without notice. You bear the risk of using it. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.