PHAT MONKEY IT

WHITE PAPER

Architecting for Agentic AI

A vendor-neutral blueprint for governing autonomous agents across Azure, AWS and on-premises estates.

Vishal Vashisht
July 2026

EXECUTIVE SUMMARY

Scaling AI agents safely requires an architecture, not just better models. This paper sets out a three-plane reference pattern – a control plane, a data plane and an application plane – for governing agents at enterprise scale. The pattern is deliberately implementation-agnostic: it maps cleanly onto Microsoft Azure, Amazon Web Services, or a self-managed on-premises estate, and can be adopted incrementally, starting with governance foundations before agent numbers grow.

01 Why agentic AI needs an architecture

Enterprises are moving from single-purpose AI features to populations of agents: autonomous software that retrieves information, reasons over it and acts on the organisation's behalf. Left unmanaged, this creates the same risk every distributed system creates when it grows without a plan – duplicated integration work, inconsistent authentication, and no reliable way to answer "who accessed what, and why".

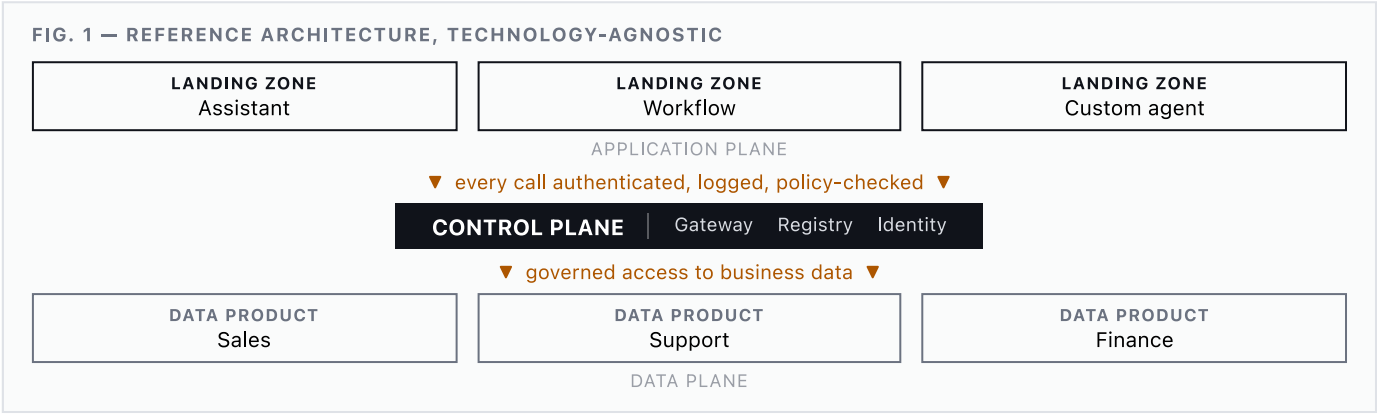
A cluster of open and vendor standards is emerging to address this: the Model Context Protocol (MCP) for tool and knowledge access, Agent-to-Agent (A2A) for cross-boundary delegation, and workload-identity patterns that give every agent a verifiable principal. None of these standards are exclusive to one cloud – which is precisely why the reference architecture below is built to be portable.

This paper generalises the hub-and-spoke reference pattern set out in Microsoft's 2026 paper, *Enterprise Architecture for Agentic AI*, into a vendor-neutral form. The estate is organised into three planes. A **control plane** provides the single governed path to models, tools and identity. A **data plane** exposes trusted, well-understood business data through standard protocols. An **application plane** hosts the agents themselves in standardised, per-workload landing zones. Each plane can be built on a different vendor's primitives without changing the pattern – including on a private cloud built on VMware or Nutanix, or on Red Hat OpenShift.

GROUNDED IN ESTABLISHED FRAMEWORKS

AWS and Azure each publish a formal Well-Architected Framework, and we map our recommendations to their security, reliability and operational-excellence pillars throughout. VMware and Nutanix express the same discipline through Validated Solutions and reference architectures rather than a single named framework, but the underlying pillars – security, reliability, cost, performance and operations – are consistent across all four.

02 The control plane: one governed doorway



Every credible implementation we have delivered shares one non-negotiable: agents never call a model, tool or dataset directly. All traffic passes through a shared gateway and a registry that records ownership, classification and lifecycle state. An agent that is not registered is treated as unauthorised, regardless of what it is trying to do.

What differs by vendor is the implementation, not the pattern. The table below maps the same three control-plane capabilities onto Azure, AWS and a self-managed estate.

Capability	Azure	AWS	On-premises / other cloud
Model & tool gateway	API Management, MCP-aware routing	Bedrock AgentCore Gateway	Self-hosted gateway (e.g. Kong) with MCP support
Tool & agent registry	API Centre + MCP inventory	AgentCore identity & gateway registry	Internal service catalogue (e.g. Backstage)
Agent identity & policy	Entra Agent ID + Azure Policy	AgentCore Identity + AgentCore Policy (Cedar)	SPIFFE/SPIRE identities + Open Policy Agent

03 The data plane: trusted, understood data

Agents are only as reliable as the data they can reach. That requires a governed lake that unifies operational and analytical data, a cataloguing layer that classifies and tracks it, and – increasingly – an explicit knowledge layer (ontologies and semantic models) so agents interpret relationships correctly instead of guessing at them from raw tables, which is a common source of hallucination.

This is the layer most exposed to vendor gravity, since data platforms are sticky. We recommend standardising the *interface* (MCP-exposed, governed data products) even where the underlying store is vendor-specific.

Capability	Azure	AWS	On-premises / other cloud
Unified data lake	Fabric OneLake (Delta format)	S3 Tables (Apache Iceberg) + Lake Formation	Self-managed lakehouse on object storage
Governance & lineage	Microsoft Purview	Glue Data Catalog + Lake Formation	Open-source catalogue (e.g. OpenMetadata)
Knowledge & semantic layer	Fabric IQ (ontologies, semantic models)	Bedrock Knowledge Bases + graph/vector store	Graph DB (e.g. Neo4j) + vector store (e.g. pgvector)

04 Platform notes: VMware, Nutanix and OpenShift

The same three planes apply just as well inside a private data centre. VMware, Nutanix and Red Hat each now ship a control-plane concept of their own for agentic workloads – the table below maps them onto the pattern.

Plane	VMware Cloud Foundation	Nutanix	Red Hat OpenShift
Control plane	VCF 9.1 zero-trust spans VMs, containers and agents; centrally managed MCP tool/server access	Nutanix Enterprise AI – Agent Gateway + Inference Management as the AI control plane	Llama Stack agentic-loop runtime with integrated MCP client; MCP server enforcing OAuth/OIDC, RBAC, audit logging
Data plane	Private AI Services: Model Gallery & Governance, managed vector database for retrieval	Nutanix Unified Storage & Data Lens, with governed Models-as-a-Service	Llama Stack vector database & RAG tooling, backed by OpenShift data services
Application plane	Agent Builder composes models, knowledge bases and tools; runs as Tanzu Kubernetes (VKS) workloads	Nutanix Kubernetes Platform hosts agents on AHV; Agent Builder layer via NVIDIA AI Enterprise	OpenShift AI serves models via vLLM/KServe; Service Mesh (Istio/Kiali) enforces inter-agent policy

None of these platforms replace the control-plane discipline above – they implement it. A private-cloud estate on VMware or Nutanix, or a Kubernetes estate on OpenShift, can sit behind the same gateway, registry and identity model as a spoke alongside Azure or AWS, rather than as a separate, ungoverned island.

05 The application plane: where agents run

Not every agent needs the same build. We consistently see three hosting patterns coexist within a single enterprise, each with a different balance of convenience and control – and each still required to consume the control plane rather than bypass it.

FIG. 2 — THREE HOSTING PATTERNS FOR AGENT WORKLOADS

No-code / low-code
Business-user agents built in a managed studio, tied to project-level guardrails.

Managed hosted runtime
Custom code, packaged and run on a session-isolated managed platform.

Self-managed containers
Full control over runtime and dependencies, still calling the shared gateway.

Pattern	Azure	AWS	On-premises / other cloud
No-code / low-code	Copilot Studio, Foundry prompt agents	Bedrock Agents (console-configured)	Internal low-code automation platform
Managed hosted runtime	Foundry Agent Service hosted agents	AgentCore Runtime (session-isolated)	Rare; typically approximated with Kubernetes
Self-managed containers	AKS / Azure Container Apps	Amazon EKS / ECS	Kubernetes with custom operators

06 Delivery approach: build in stages

We do not recommend a big-bang platform build. Stand up a minimal control plane – gateway, registry and identity – before writing a single agent. Onboard one representative workload per pattern above: one interactive assistant, one data-centric pipeline, one multi-step workflow. Only then scale, through templates, automated policy checks and lifecycle tooling, rather than repeated bespoke reviews.

The sequence matters more than the vendor: a governed pilot on any cloud outperforms an ungoverned rollout on the “right” one.

> **A PHAT MONKEY PERSPECTIVE**
>
> As an independent, boutique consultancy with senior engineers across AWS, Azure, VMware and Nutanix estates, our advice starts from the governance pattern, not the platform. Our London-based team combines deep technical skill with a dedication to hands-on, personal client service, helping organisations design and stand up the control plane first, then choose the infrastructure – public cloud, private cloud or hybrid – that best fits the workload, cost and regulatory posture at hand.

Sources: Microsoft, ‘Enterprise Architecture for Agentic AI’ (2026); AWS Well-Architected Framework and AWS Prescriptive Guidance on agentic AI; Microsoft Azure Well-Architected Framework; Broadcom/VMware Cloud Foundation 9.1 and Private AI Services documentation; Nutanix Agentic AI and Nutanix Enterprise AI product documentation; Red Hat OpenShift AI and Llama Stack documentation. Prepared by Phat Monkey IT Ltd for briefing purposes; not affiliated with Microsoft, AWS, Broadcom/VMware, Nutanix or Red Hat.