Hybrid-Cloud AI Implementation Playbook

A Practical, Phased Roadmap for Deploying Agentic AI Across the Enterprise

Digital Enterprise Architecture Advisors (DEAA)

Overview

This playbook provides a **practical, phased implementation roadmap** for deploying enterprise AI systems across hybrid-cloud environments. It operationalizes the Hybrid-Cloud AI Reference Architecture Blueprint through five phases: hybrid foundation, AI control plane, MCP integration, agent development, and runtime governance. Each phase includes detailed steps, governance checkpoints, and architectural considerations to ensure secure, compliant, and scalable deployment of agentic AI systems.

Author

Christian Kobsa
Strategic Enterprise Architect
Digital Enterprise Architecture Advisors

Version

v1.0 — 2026

A practical roadmap from **current state** to a **fully agentic, hybrid-cloud AI enterprise architecture**.

Phase 1 — Foundation: Hybrid connectivity, identity, data, and DevSecOps

Objective:
Establish the **technical and governance foundation** for AI and agents.

Step 1.1 — Establish hybrid connectivity

- Implement **VPN, Direct Connect, and/or SD-WAN** between:
  - On-prem data centers
  - Private cloud (OpenShift, VMware, IBM PowerVS)
  - Public clouds (IBM Cloud, AWS, Azure, GCP)
- Enforce **network segmentation** and **zero-trust access**.

Step 1.2 — Implement identity federation

- Integrate **OIDC, IAM, LDAP** across environments.
- Define **identity propagation patterns** for:
  - Users → agents
  - Agents → tools (via MCP)
- Ensure **least-privilege access** and **role-based controls**.

Step 1.3 — Deploy lakehouse and vector infrastructure

- Stand up a **unified data lakehouse** (e.g., Databricks, Snowflake, or equivalent).
- Implement:
  - **Vector indexes** for RAG
  - **ML feature stores**
  - **Batch and streaming pipelines**
- Embed **data governance**:

- Lineage
- Access control
- Quality rules
- Synthetic data policies

Step 1.4 — Stand up DevSecOps pipelines

- Extend existing CI/CD to **AI/ML and agent workloads**:
  - Infrastructure as code
  - Model deployment pipelines
  - Policy-as-code
- Integrate:
  - **Security scanning**
  - **Compliance checks**
  - **Audit logging**
- Treat DevSecOps as the **execution layer** for **continuous assurance**.

Phase 2 — AI control plane: LLM gateway and guardrails

Objective:
Create a **central AI control plane** for all model interactions.

Step 2.1 — Deploy LLM gateway

- Implement an **AI/LLM gateway** that:
- Routes requests to **cloud, on-prem, and edge models**
- Enforces **policies and guardrails**
- Propagates **identity and context**
- Logs **all interactions**

Step 2.2 — Integrate guardrails

- Configure:
    - **Safety filters**
    - **Content policies**
    - **Rate limits and throttling**
- Implement **LLM-as-a-Judge** and **policy checks** for:
- Safety
- Compliance
- Acceptable agency

Step 2.3 — Configure logging and audit

- Centralize:
    - **Request/response logs**
    - **Model selection decisions**
    - **Policy enforcement outcomes**
- Ensure logs are:
- **Immutable**
- **Searchable**
- **Linked to identities and agents**

Phase 3 — MCP integration: Tooling and enterprise systems

Objective:
Expose enterprise systems as **governed tools** via MCP.

Step 3.1 — Deploy MCP servers across cloud and on-prem systems

- Implement MCP servers for:
- **IBM Cloud**
- **IBM Storage Insights**

- **IBM PowerVS**
- **IBM Z** (e.g., watsonx Assistant for Z)
- **IBM TLS**
- Other enterprise systems (databases, IT tools, business apps)

Step 3.2 — Standardize tool schemas

- Define **typed schemas** for:
    - Inputs
    - Outputs
    - Error conditions
- Version and document each tool.

Step 3.3 — Build enterprise tool catalog

- Maintain a **catalog** with:
    - Tool descriptions
    - Owners
    - Risk posture
    - Allowed agents and use cases
- Link tools to:
- **Policies**
- **Audit requirements**
- **Environments** (dev, test, prod)

Phase 4 — Agent development: ADLC in practice

Objective:
Design, build, evaluate, and certify agents using the **Agent Development Lifecycle**.

Step 4.1 — Define agent personas and authority scopes

- For each agent type (IT ops, security, customer service, research, workflow automation):
- Define **goals and responsibilities**
- Specify **authority bounds** (what they can and cannot do)
- Design **human-in-the-loop** paths and **escalation rules**
- Define **kill switch** conditions

Step 4.2 — Implement memory and planning

- Implement:
    - **Short-term memory** (per session/task)
    - **Long-term memory** (episodic, knowledge)
- Add **planning capabilities**:
    - ReAct
    - Reflexion
    - Hierarchical planning
- Ensure memory is **governed** (retention, access, poisoning defenses).

Step 4.3 — Integrate tools via MCP

- Connect agents to **MCP servers**:
    - IT operations tools
    - Security tools
    - Business applications
    - Data platforms
- Enforce:
- **Least-privilege tool access**
- **Policy checks** at the gateway
- **Audit logging** for all tool calls

Step 4.4 — Run evaluation suites (evaluation-first)

- Design **behavioral benchmarks** per agent:
    - Task success
    - Trajectory correctness
    - Safety and compliance
- Use:
    - **LLM-as-a-Judge**
    - **Human-in-the-loop review**
- Measure:
- **Hallucination rates**
- **Bias and fairness**
- **Drift over time**

Step 4.5 — Certify agents

- Require:
    - **Evaluation results**
    - **Red teaming reports**
    - **Compliance checklists**
- Classify:
    - **Risk posture** (low/medium/high)
- Approve agents into a **governed catalog** before production.

Phase 5 — Deployment & governance: Runtime operations

Objective:
Deploy agents safely and operate them under **continuous governance**.

Step 5.1 — Deploy with kill switches and feature flags

- Use:
    - **Feature flags** for gradual rollout
    - **Kill switches** at:
    - Agent level
    - Tool level
    - Environment level
- Roll out:
- **Progressively** (canary, phased)
- With **rollback strategies**

Step 5.2 — Monitor reasoning traces and tool usage

- Collect:
    - **Reasoning traces**
    - **Tool call logs**
    - **Latency and cost metrics**
- Detect:
- **Behavioral drift**
- **Performance regressions**
- **Anomalous tool usage**

Step 5.3 — Continuously optimize

- Tune:
    - **Prompts**
    - **Memory policies**
    - **Tool selection**
    - **Model routing**
- Use feedback from:

- **Users**
- **LLM-aaJ**
- **Operational metrics**

Step 5.4 — Maintain continuous governance

- Align with **NIST AI RMF**:
  - Govern
  - Map
  - Measure
  - Manage
- Maintain:
  - **Audit trails**
  - **Compliance evidence**
  - **Risk scoring**
- Treat governance as **continuous**, not periodic.

6. Organizational implications

6.1 Data engineering becomes strategic

Data engineers evolve into:

- **Platform architects**
- **AI pipeline owners**
- **Governance enforcers**
- **Multimodal data specialists**

They move from **pipeline plumbing** to **designing reusable data platforms, governance, and oversight**, enabling AI at scale.

6.2 IT operations becomes autonomous

Agentic AI enables:

- **Self-healing infrastructure**
- **Automated incident response**
- **Predictive maintenance**
- **Closed-loop remediation**

IT operations shifts from **manual, reactive** work to **autonomous, agent-driven** operations.

6.3 Governance becomes continuous

Compliance shifts from **periodic audits** to:

- **Real-time monitoring**
- **Automated evidence collection**
- **Continuous risk scoring**

Governance is embedded into:

- **Data**
- **Models**
- **Tools**
- **Agents**
- **Operations**

7. Conclusion

A corporate AI-based enterprise architecture must be:

- **Hybrid-cloud native**
- **Agentic AI-ready**
- **MCP-integrated**
- **Secure-by-design**
- **Evaluation-first**

- **Governed end-to-end**

This architecture transforms enterprises from **reactive, manual operations** into **autonomous, intelligent, and resilient systems** capable of **scaling AI safely and effectively** across hybrid-cloud environments.