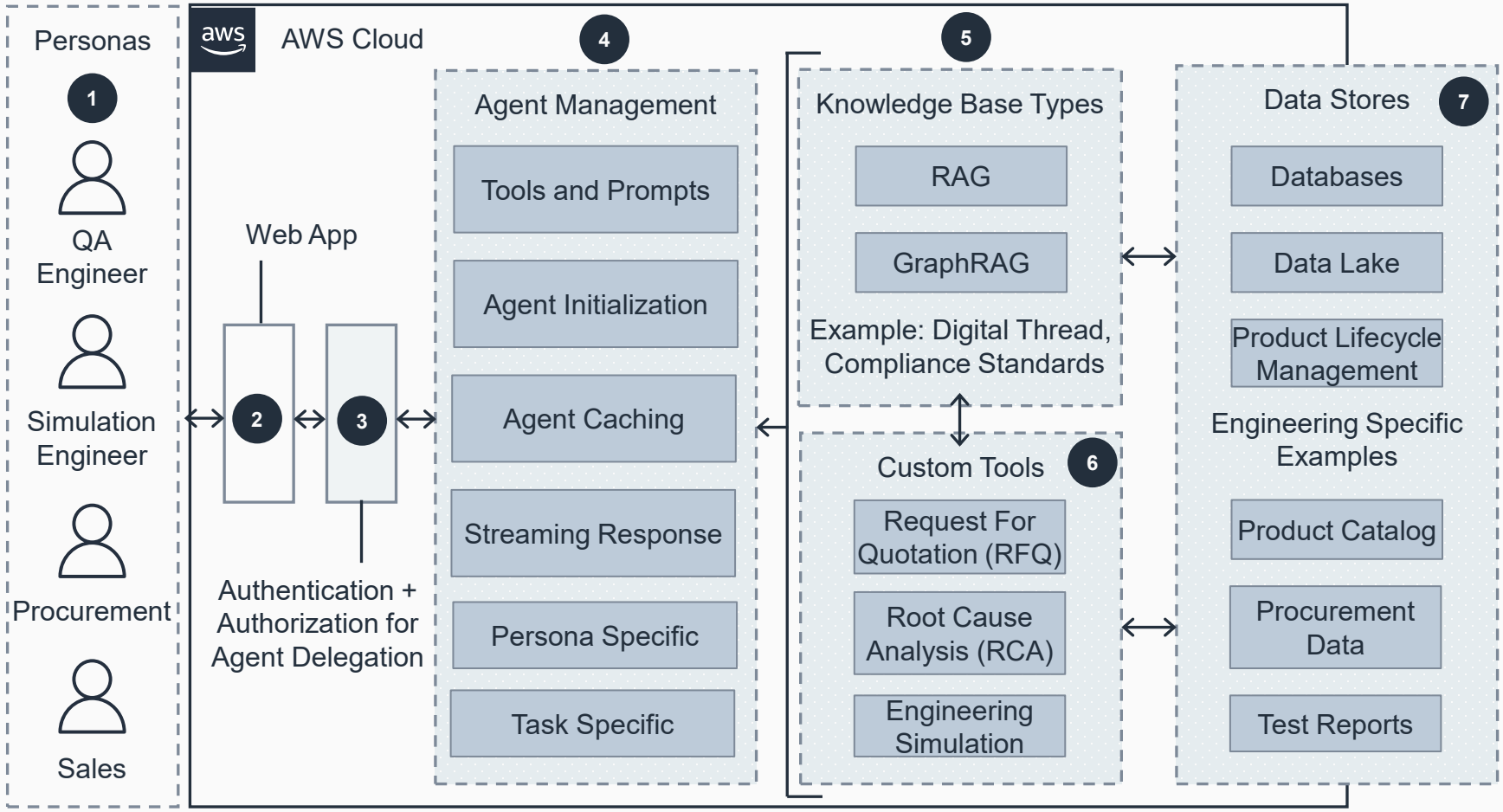
Guidance for Building Agentic AI-Powered Knowledge Assistant on AWS

Diagram 1

This diagram illustrates a flexible agentic AI framework on AWS that dynamically assigns specialized agents based on user identity and needs. The system authenticates users and provisions appropriate agent capabilities, maintaining security boundaries while enabling modular addition of new agents and tools for specific engineering domains.



1. Each persona has a set of agents to help perform their respective tasks.
2. Users engage with the system through a web application that processes initial requests.
3. Verify identity and permissions through the system's authentication process. Based on the verified role, the system determines which specialized AI agent capabilities to activate for the session.
4. Coordinate specialized AI functions through the agent management layer, which initializes agents, caches information, streams responses, and adapts to the persona and task requirements.
5. Access relevant information sources including Retrieval Augmented Generation (RAG) for document retrieval, GraphRAG for relationship analysis, and specialized knowledge bases like Digital Thread and Compliance Standards documentation. These sources provide context-aware responses tailored to the engineering domain.
6. Execute custom engineering workflows through intelligent agents that access third-party Application Programming Interfaces (APIs), query data stores, and perform specialized logic based on requirements.
7. Store and organize engineering data across multiple systems including databases, data lakes, and product lifecycle management systems for continuous access and analysis.

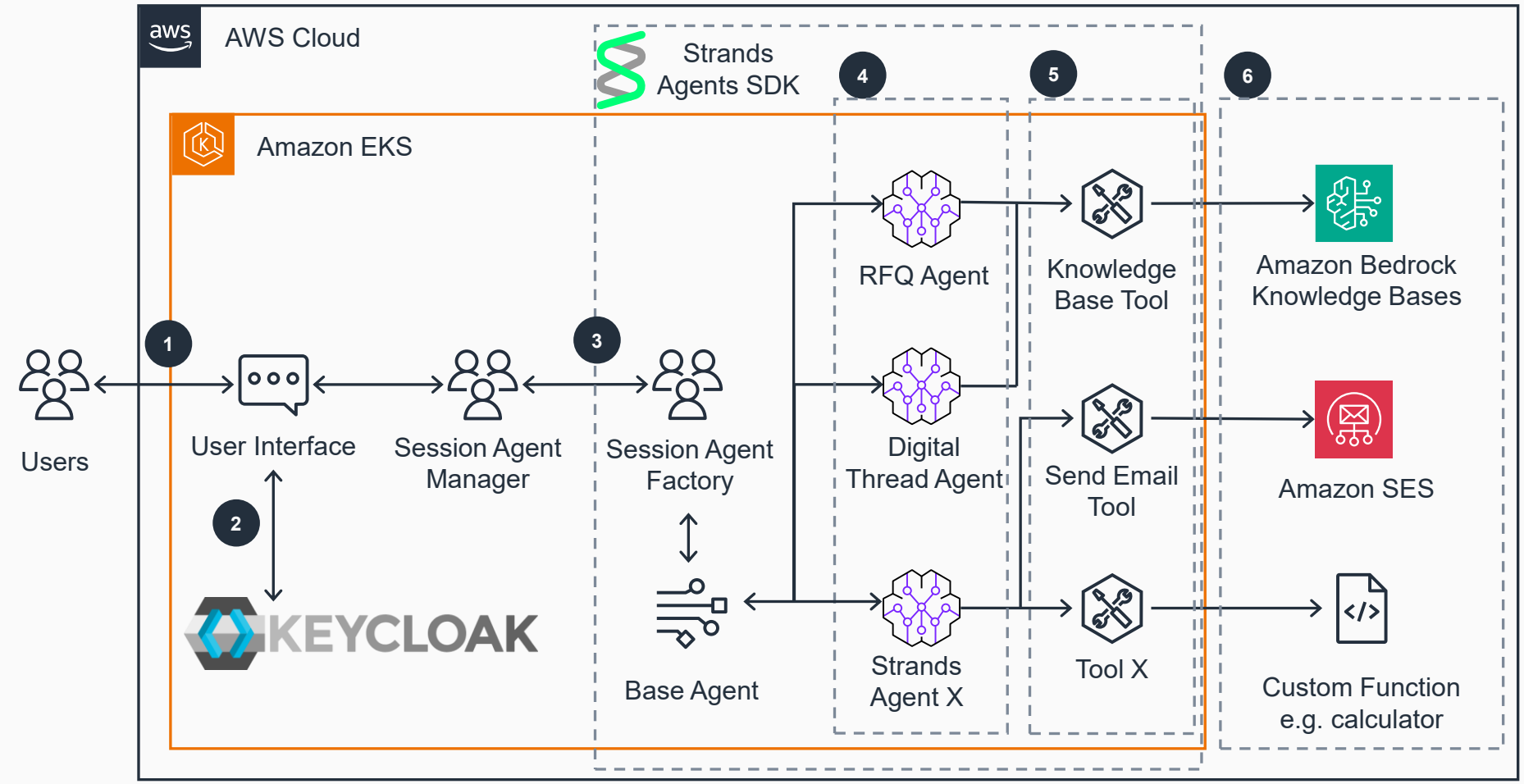
Reviewed for technical accuracy November 17, 2025

AWS Reference Architecture

Guidance for Building Agentic AI-Powered Knowledge Assistant on AWS

Diagram 2

This architecture diagram implements key components from the previous framework: Authentication (Section 2) with Keycloak, Agent Management (Section 3) via the Session Agent Manager, specialized agent routing (Section 4) using AWS Strands Agents SDK, and Custom Tool integration (Section 6) including RAG operations with Bedrock Knowledge Bases and RFQ processing - all hosted on Amazon Elastic Kubernetes Service to provide a secure, flexible framework for engineering R&D.



1. Submit engineering queries through the chat interface hosted on **Amazon Elastic Kubernetes Service (Amazon EKS)**.
2. Authenticate via Keycloak. The system routes requests to the session agent manager for orchestration.
3. Session agent manager defines role-based access to agents. The manager initializes and caches each agent for the session.
4. Route requests to agents built using AWS Strands Agents SDK. The SDK enables developers to create custom agents (such as RFQ agents or Digital Thread agents) that can be specialized for different engineering workflows. The SDK provides the building blocks and primitives needed for multi-agent implementations.
5. Agents leverage tools that define code-based functionality. Available tools include Knowledge Base Tool for information retrieval, Send Email Tool for **Amazon Simple Email Service (Amazon SES)** communication, and customizable implementations supporting multiple agents.
6. Tools perform retrieval augmented generation (RAG) with **Amazon Bedrock** Knowledge Bases, send RFQ emails via SES, and execute custom functions to complete queries.

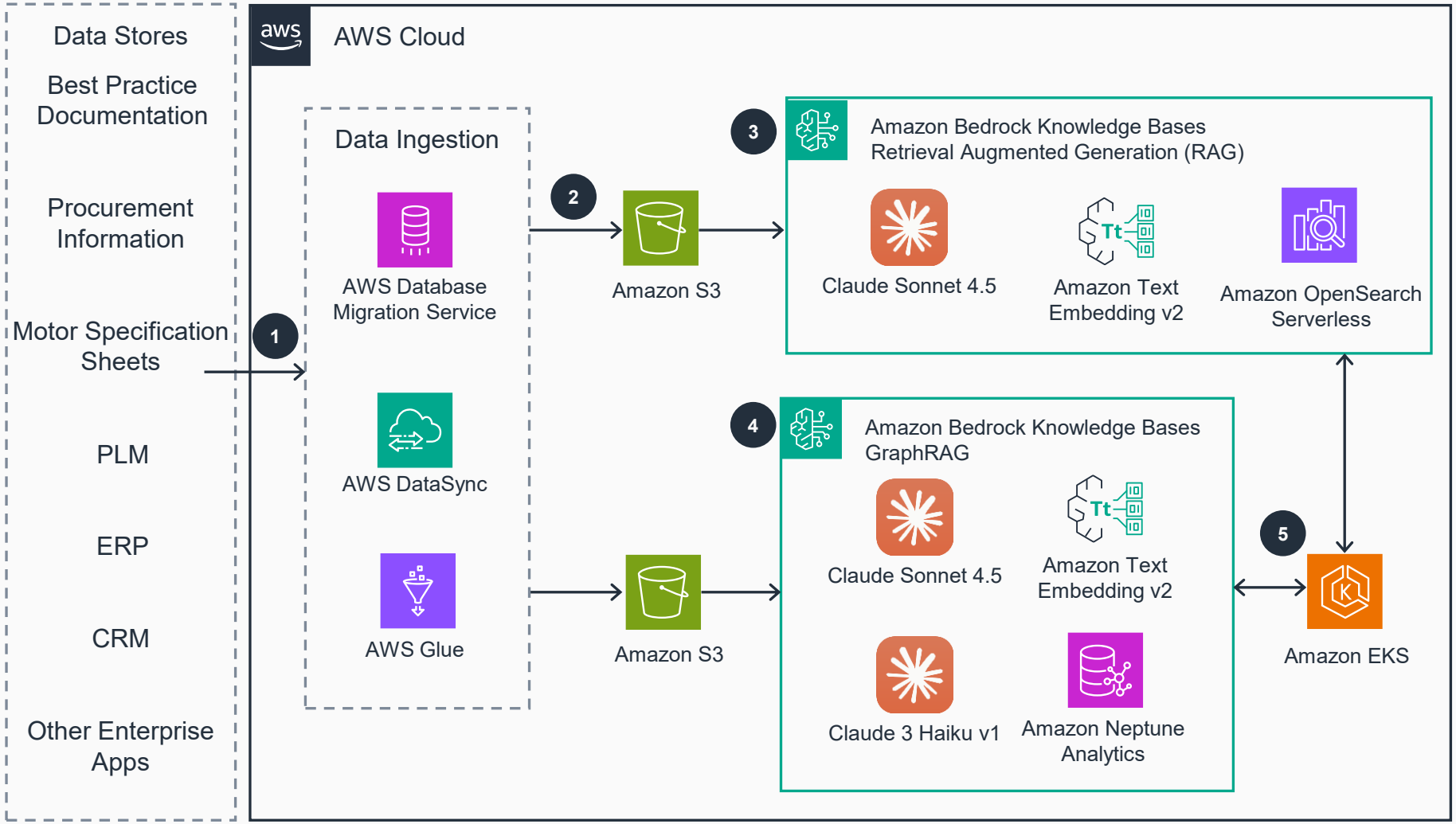
Reviewed for technical accuracy November 17, 2025

AWS Reference Architecture

Guidance for Building Agentic AI-Powered Knowledge Assistant on AWS

Diagram 3

This architecture diagram shows a knowledge base for retrieval augmented generation (RAG) and GraphRAG pipeline.



1. Ingest data from identified sources using **AWS Database Migration Service (AWS DMS)**, **AWS DataSync** for large datasets, and **AWS Glue** for ETL jobs into **Amazon Simple Storage Service (Amazon S3)**.

2. Structure your **Amazon S3** data with a purpose-driven organization strategy. Create distinct buckets that align with specific business functions and user needs. This thoughtful organization improves retrieval accuracy and maintains appropriate context for different user groups' queries.

3. Configure **Amazon Bedrock** Knowledge Bases with **Amazon OpenSearch Serverless** to ingest data from **Amazon S3**. While **Amazon Bedrock** provides access to various foundation models through a unified API, we've used Amazon Titan Embeddings for creating vector embeddings, enabling efficient similarity search ideal for unstructured data like compliance documents and manuals.

4. Configure **Amazon Bedrock** Knowledge Bases with **Amazon Neptune** Analytics to ingest relational data from **Amazon S3**. This setup excels at handling complex, interconnected datasets like supply chain networks, enabling queries that understand relationships between suppliers, materials, and distribution channels.

5. Query the **Amazon Bedrock** Knowledge Base through API calls to ground LLM responses with your organizational data for factually accurate results.

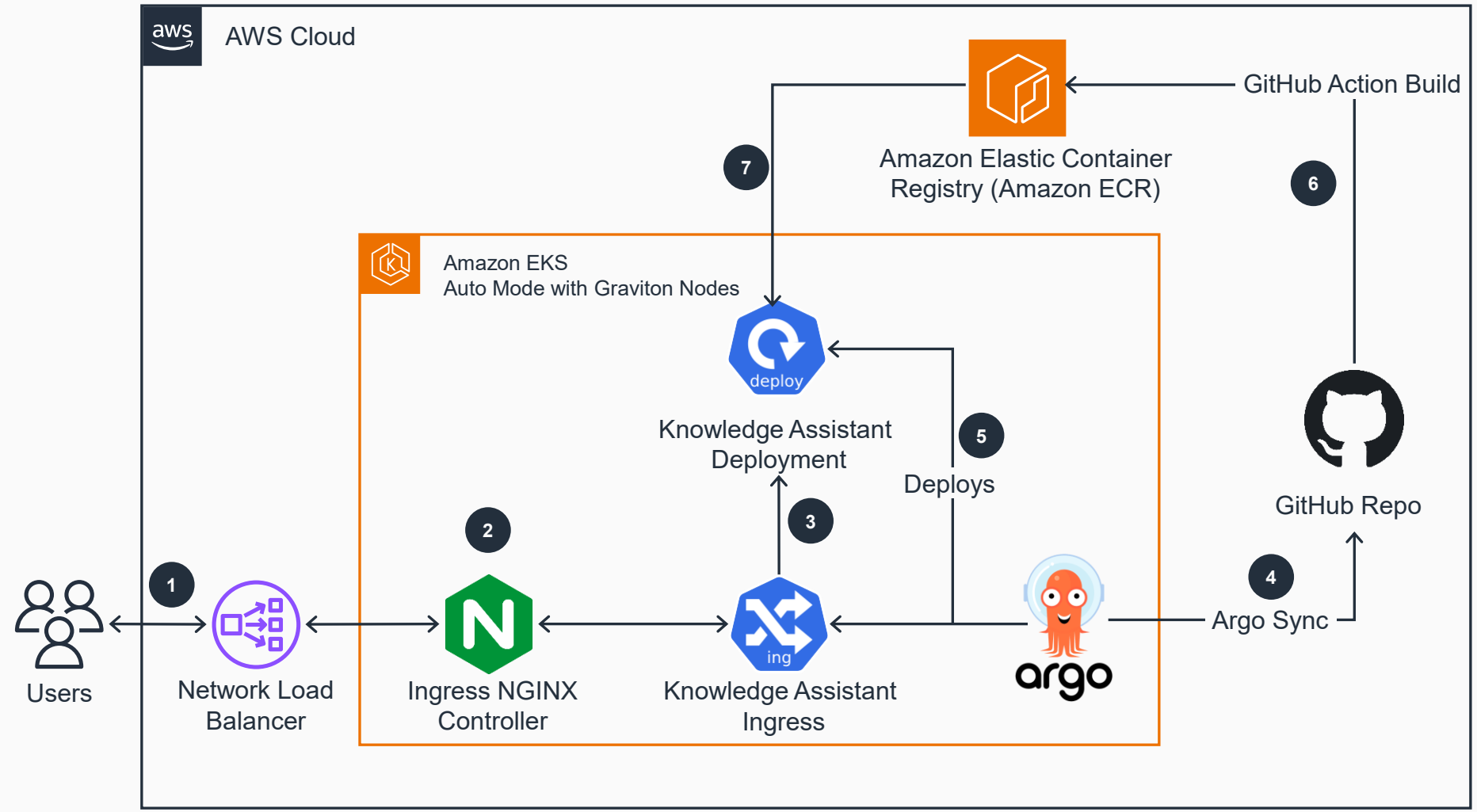
Reviewed for technical accuracy November 17, 2025

AWS Reference Architecture

Guidance for Building Agentic AI-Powered Knowledge Assistant on AWS

Diagram 4

This architecture diagram shows continuous integration and continuous deployment diagram of the end-to-end solution.



1. Access the application through the Network Load Balancer (NLB) created by Ingress NGINX for traffic distribution.
2. Ingress NGINX routes incoming traffic to the correct ingress using path-based routing rules.
3. Knowledge Assistant Ingress directs traffic to the appropriate endpoint within the Knowledge Assistant application.
4. Argo CD syncs Knowledge Assistant application manifests from the GitHub repository, serving as the source-of-truth for GitOps deployment
5. Argo CD deploys the synchronized manifests from the source-of-truth onto the **Amazon Elastic Kubernetes Service (Amazon EKS)** cluster.
6. GitHub Actions builds the Knowledge Assistant Docker image and pushes it to **Amazon Elastic Container Registry (Amazon ECR**) upon each release.
7. Knowledge Assistant deployment pulls the latest image from **Amazon ECR** for container orchestration.

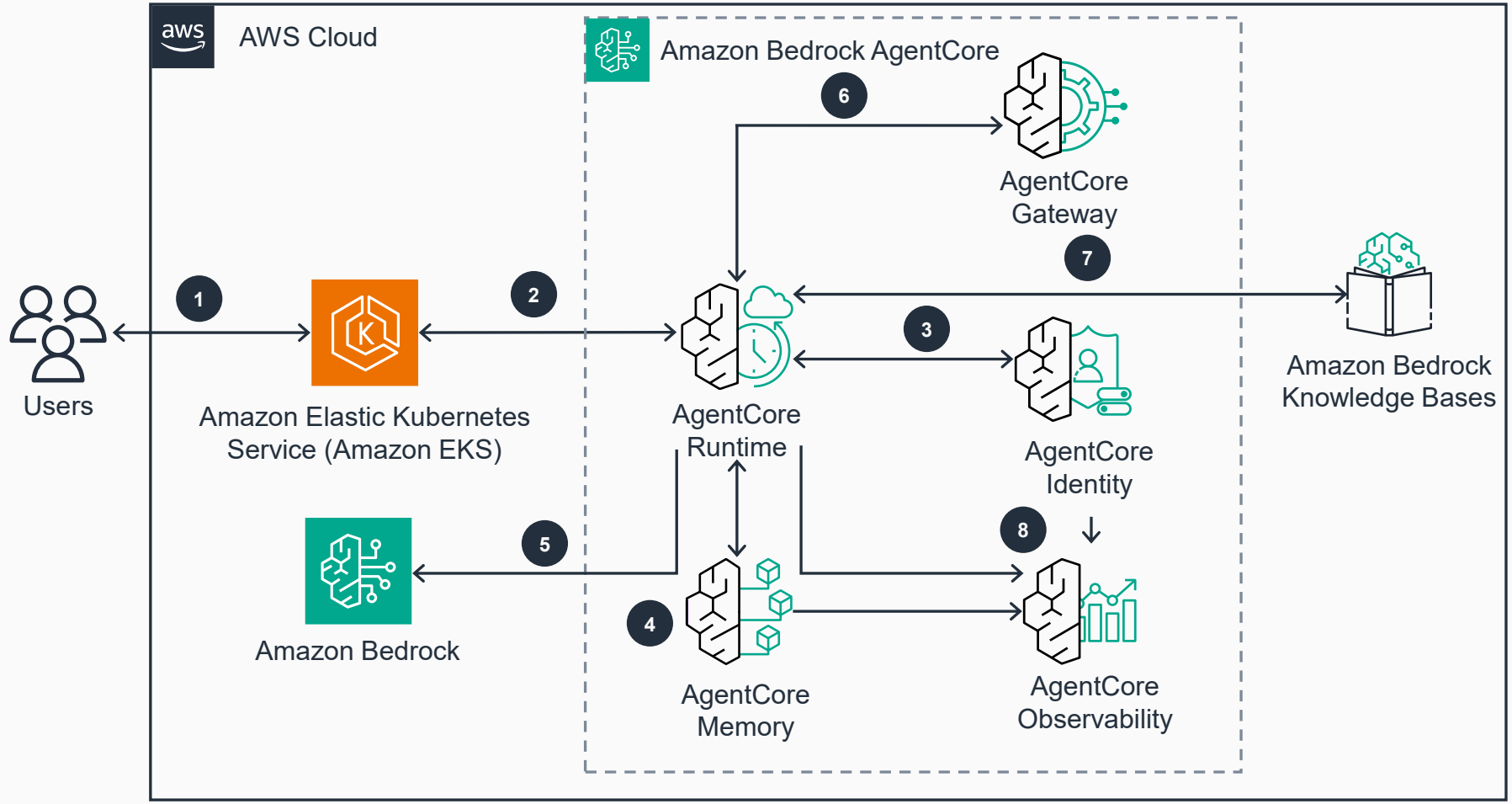
Reviewed for technical accuracy November 17, 2025

AWS Reference Architecture

Guidance for Building Agentic AI-Powered Knowledge Assistant on AWS

Diagram 5

This architecture diagram shows alternative agent hosting on Amazon Bedrock AgentCore. Deploy secure, scalable AI agents on AWS with AgentCore's comprehensive set of enterprise-grade services, enabling complex workflows across tools and data sources while eliminating infrastructure management overhead.



1. Submit engineering queries through the chat interface hosted on **Amazon Elastic Kubernetes Service (Amazon EKS)**.
2. Execute agent code, tools, and instructions in **Amazon Bedrock AgentCore** Runtime's serverless environment, supporting multiple frameworks and 8-hour sessions.
3. Secure agent operations with **Amazon Bedrock AgentCore** Identity, managing authentication and access controls across all interactions.
4. Build context-aware agents with **Amazon Bedrock AgentCore** Memory, maintaining both short-term and long-term knowledge across interactions.
5. Access **Amazon Bedrock** for foundation models, enabling flexible use of various LLMs through a unified API.
6. Transform REST APIs into Model Context Protocol (MCP) servers through **Amazon Bedrock AgentCore** Gateway, enabling reusable tool sharing across agents.
7. Connect to third-party tools like **Amazon Bedrock** Knowledge Bases to enrich context and perform tasks.
8. Monitor agent performance through **Amazon Bedrock AgentCore** Observability, tracking key metrics and ensuring operational excellence.

Reviewed for technical accuracy November 17, 2025

AWS Reference Architecture