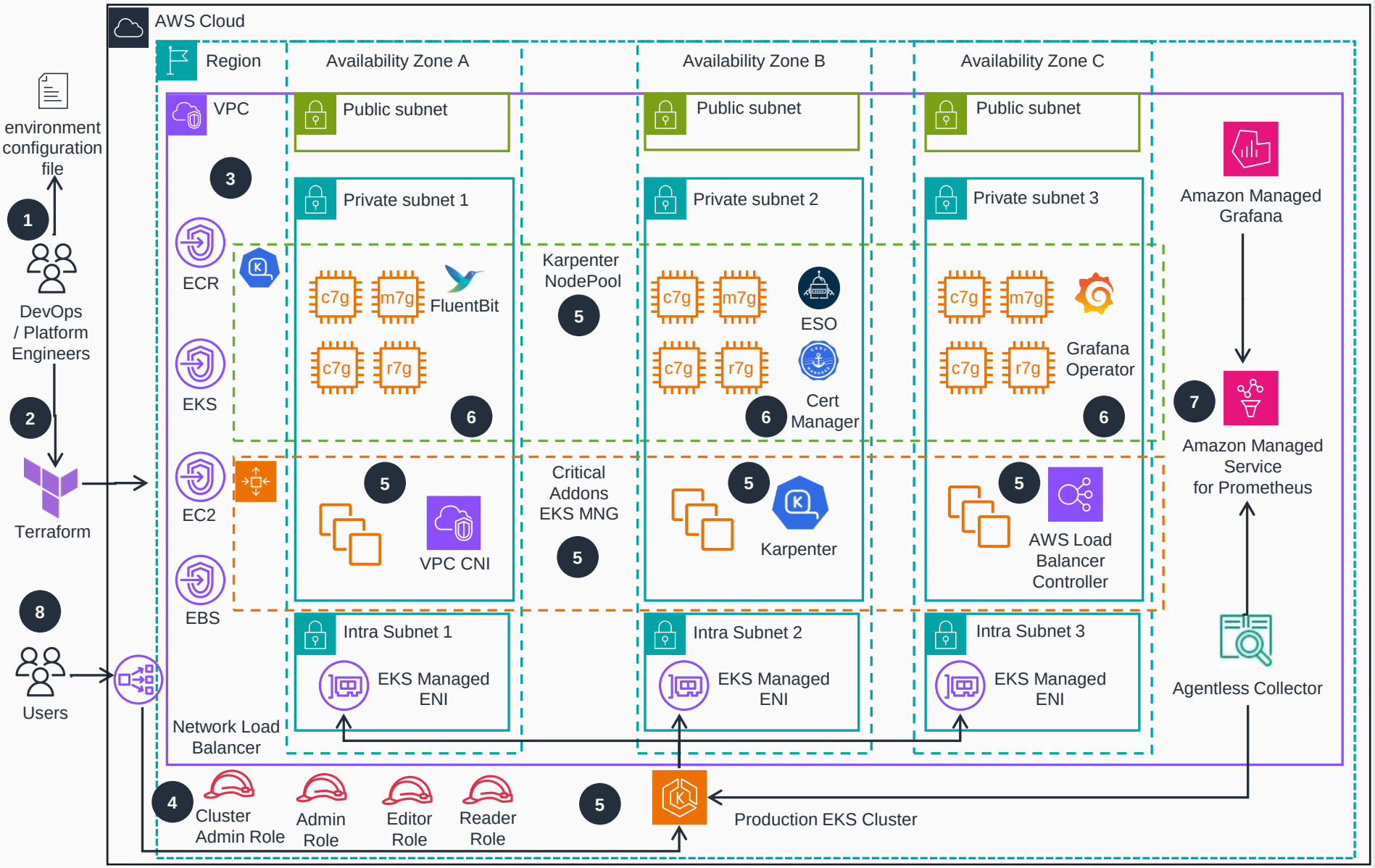
Guidance for Scalable Model Inference and Agentic AI on Amazon EKS

This diagram shows how to provision an Amazon Elastic Kubernetes Service (EKS) cluster with best practices configuration and critical add-ons for AI workloads



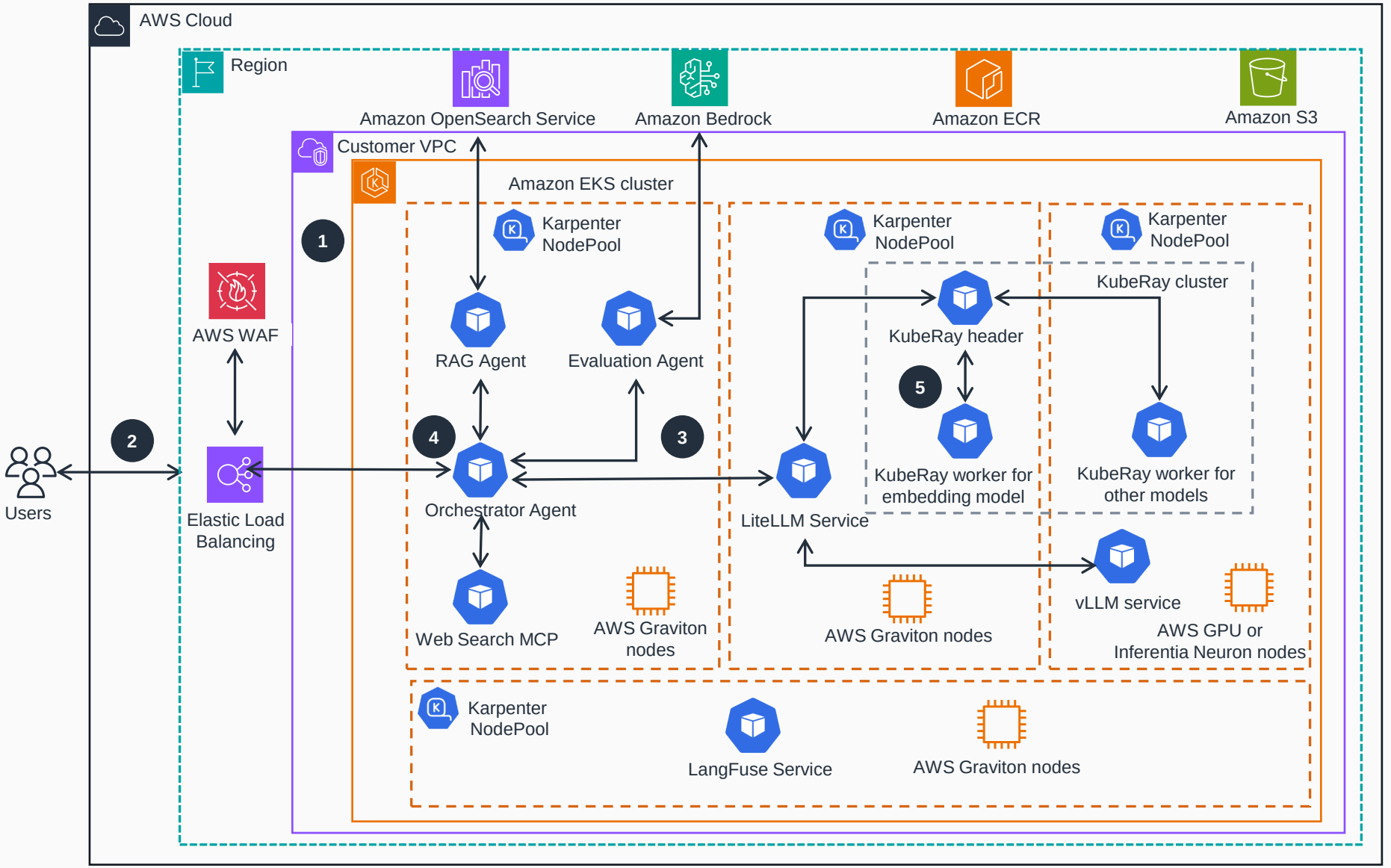
1. DevOps engineer defines a per-environment Terraform variable file that control all of the environment-specific configuration. This configuration file is used in all steps of deployment process by all IaC configurations to create different EKS (Amazon Elastic Kubernetes Service) environments
2. DevOps engineer applies the environment configuration using Terraform following the deployment process defined in the guidance.
3. An Amazon Virtual Private Cloud (VPC) is provisioned and configured based on specified configuration. According to best practices for Reliability, 3 Availability zones (AZs) are configured with corresponding VPC Endpoints to provide access to resources deployed in private VPC.
4. User facing Identity and Access Management (IAM) roles (Cluster Admin, Admin, Editor and Reader) are created for various access levels to EKS cluster resources, as recommended in Kubernetes security best practices
5. EKS cluster is provisioned with Managed Nodes Groups that run critical cluster add-ons (CoreDNS, AWS Load Balancer Controller and Karpenter) on its compute node instances. Karpenter will manage compute capacity for other add-ons, as well as applications that will be deployed by user while prioritizing price-performance efficient AWS Graviton instances
6. Other important EKS add-ons (Cert Manager, FluentBit, Grafana Operator) are deployed based on the configurations defined in the environment Terraform configuration file (*see Step1 above*).
7. AWS Managed Observability stack is deployed, including Amazon Managed Service for Prometheus (AMP), AWS Managed collector for Amazon EKS, and Amazon Managed Grafana (AMG)
8. Users can access Amazon EKS cluster(s) with best practice add-ons, optionally configured Observability stack and RBAC based security mapped to IAM roles for workload deployments using Kubernetes API that is exposed via Network Load Balancer

Reviewed for technical accuracy October 2, 2025

AWS Reference Architecture

Guidance for Scalable Model Inference and Agentic AI on Amazon EKS

This architecture diagram demonstrates how to deploy ML Models and MCP server and agent on EKS cluster, provide unified model inference API Gateway, and implement Agentic AI capabilities to enable models' interactions.



1. Guidance workloads are deployed into an Amazon EKS cluster, configured for application readiness with compute plane managed by Karpenter auto-scaler. This setup efficiently auto-scales AWS Graviton, GPU and AWS Inferentia based compute nodes across multiple Availability Zones (AZs), ensuring robust infrastructure distribution and high availability of various Kubernetes services.

2. User interaction starts with an authentication to the EKS cluster via IAM. Application HTTP(s) requests are directed to an exposed endpoint, managed by Elastic Load Balancing and protected by AWS Web Application Firewall (WAF). This endpoint URL serves as the gateway for all user queries and ensures balanced distribution of incoming traffic with consistent accessibility.

3. The Orchestrator agent, powered by Strands Agent SDK, serves as the central workflow coordination hub. It processes incoming queries by connecting with reasoning models supported by LiteLLM and vLLM services, analysing the requests, and determining the appropriate workflow and tools needed for response generation. LiteLLM functions as a unified API gateway proxy for model management, providing centralized security controls and standardized access to both embedding and reasoning models.

4. Orchestrator agent delegates control to RAG agent to verify knowledge base validity to initiate Knowledge validation. When updates are needed, the RAG agent initiates the process of embedding new information into the Amazon OpenSearch cluster, ensuring the Knowledge Base remains current and comprehensive.

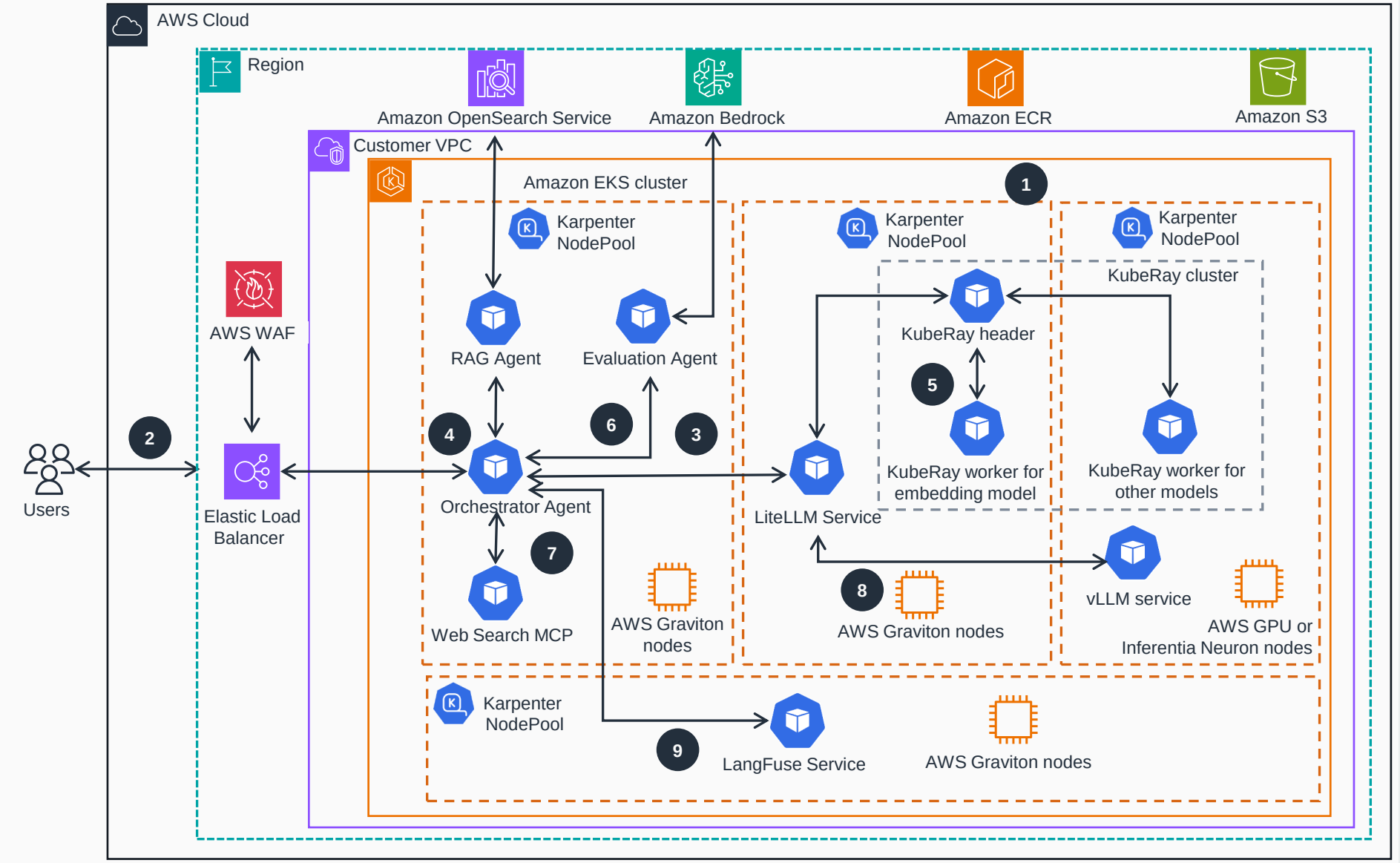
5. KubeRay cluster handles the embedding process where the Ray header dynamically manages cluster worker node scaling based on resource demands. These worker nodes execute the embedding process through the llamacpp framework, while the RAG agent simultaneously embeds user questions and searches for relevant information with the OpenSearch cluster.

Reviewed for technical accuracy October 2, 2025

AWS Reference Architecture

Guidance for Scalable Model Inference and Agentic AI on Amazon EKS

This architecture diagram demonstrates how to deploy ML Models and MCP server and agent on EKS cluster, provide unified model inference API Gateway, and implement Agentic AI capabilities to enable models' interactions.



6 Evaluation agent performs Response Quality assurance which leverages models hosted in Amazon Bedrock. This agent implements a feedback-based correction loop using RAGAS metrics to assess response quality and provides relevancy scores to the orchestrator agent for decision-making purposes.

7 When the RAG agent's responses receive relevancy scores below the configured threshold, the Orchestrator agent initiates a web search process. It retrieves the Tavily web search API tool from the Web search Model context Protocol (MCP) server and performs dynamic queries to obtain supplementary or corrective information.

8 Models based on vLLM framework running on GPU (or Inferentia) EKS compute nodes generate final responses which are returned via LiteLLM gateway to the Orchestrator agent. The reasoning model processes the aggregated information, including both Knowledge Base data and Web search results when applicable, refining and rephrasing the content to create a coherent and accurate response for the user's prompt.

9 Throughout this entire process, the Orchestrator agent maintains comprehensive interaction tracking. It automatically traces all agent activities and communications, with the resulting metrics being visualized via the LangFuse service, providing transparency and monitoring of the system's operations. Additional EKS infrastructure level observability is available via previously deployed and configured Prometheus /Grafana monitoring stack.

Reviewed for technical accuracy October 2, 2025

AWS Reference Architecture