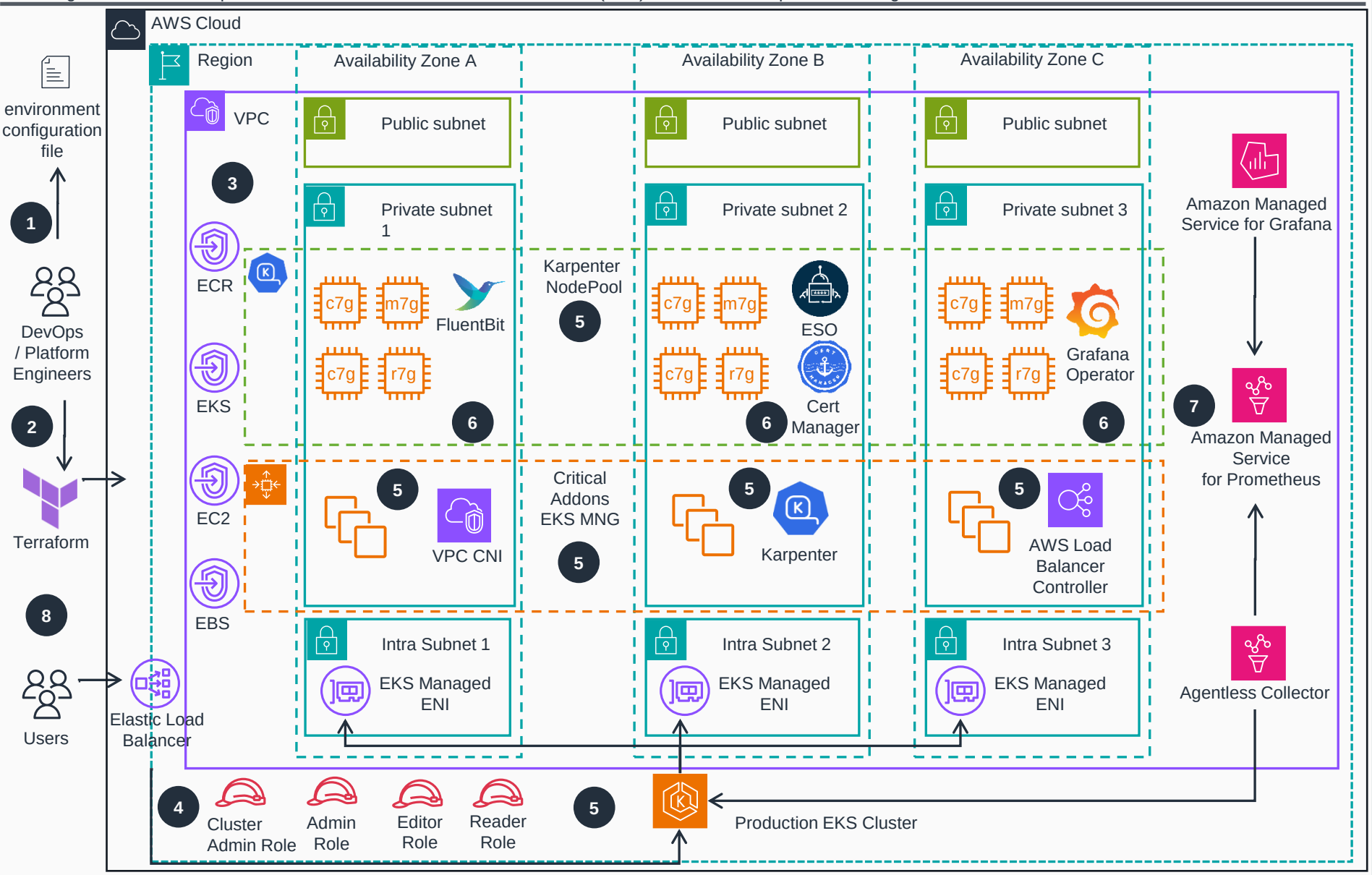
Guidance for Troubleshooting of Amazon EKS using Agentic AI workflow on AWS

Provision EKS

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



1. DevOps engineer defines a per-environment **Terraform variable file** that controls all environment-specific configuration. This configuration file is used in all steps of deployment process by all IaC configurations to provision **Amazon Elastic Kubernetes Service (Amazon EKS)** environments
2. DevOps engineer applies the environment configuration using Terraform following the deployment process defined in the guidance.
3. An **Amazon Virtual Private Network** (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 etc.) are created for various access levels to **Amazon EKS** cluster resources, as recommended in Kubernetes security best practices
5. **Amazon EKS** cluster is provisioned with Managed Nodes Groups that run critical cluster add-ons (CoreDNS, AWS Load Balancer Controller and **Karpenter** auto-scaler) on its compute node instances. Karpenter will manage compute capacity for other add-ons, as well as applications that will be deployed by user
6. Other important **Amazon EKS** add-ons (Cert Manager, FluentBit, Grafana Operator etc.) are deployed based on the configurations defined in the environment Terraform configuration file (*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 Service for 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 **AWS Network Load Balancer**

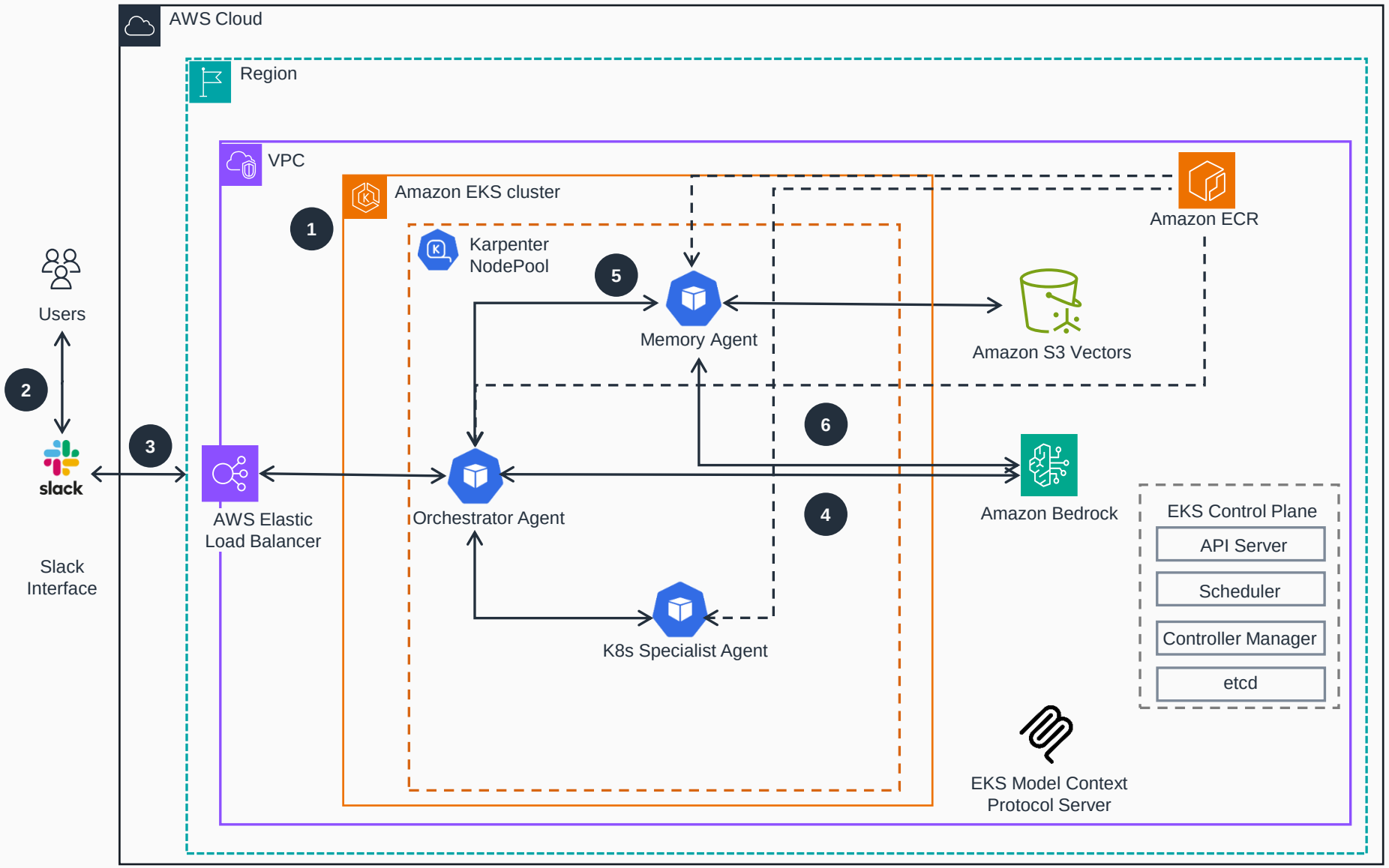
Reviewed for technical accuracy Jan,12 2026

AWS Reference Architecture

Guidance for Troubleshooting of Amazon EKS using Agentic AI workflow on AWS

Agentic AI Workflow

This architecture diagram shows troubleshooting Agentic AI workflow working with real-time EKS cluster data and integrated with AWS AI services and Slack



1. Guidance workloads are deployed into an **Amazon EKS** cluster, configured for application readiness with compute plane managed by **Karpenter** auto-scaler, as shown in Diagram 1

2. Users (DevOps engineers, SREs, developers) who encounter Kubernetes (K8s) issues send troubleshooting requests through designated Slack channel integrated with K8s Troubleshooting AI Agent. Its components are running as containers on the **Amazon EKS** cluster deployed from previously built images and hosted in **Elastic Container registry (ECR)** via Helm charts.

3. Designated Slack receives user messages via **AWS Elastic Load Balancer** and establishes a WebSocket connection (Socket Mode) to the Orchestrator agent running on the **Amazon EKS** cluster

4. Orchestrator agent receives users' message and calls **Amazon Nova Micro** via **Amazon Bedrock**, a fully managed service for foundation models, to determine whether the message requires K8s troubleshooting. If an issue is classified as K8s-related, the Orchestrator agent initiates a workflow by delegating tasks to specialized agents while maintaining overall user session context.

5. Orchestrator agent invokes the Memory agent, which connects to **Amazon S3 Vectors** based knowledge base to search for similar troubleshooting cases for precise classification

6. The Memory agent invokes **Amazon Titan** Text Embeddings via **Amazon Bedrock** to generate semantic embeddings and perform vector similarity matching against the shared **Amazon Simple Storage Service (Amazon S3)** Vectors knowledge base

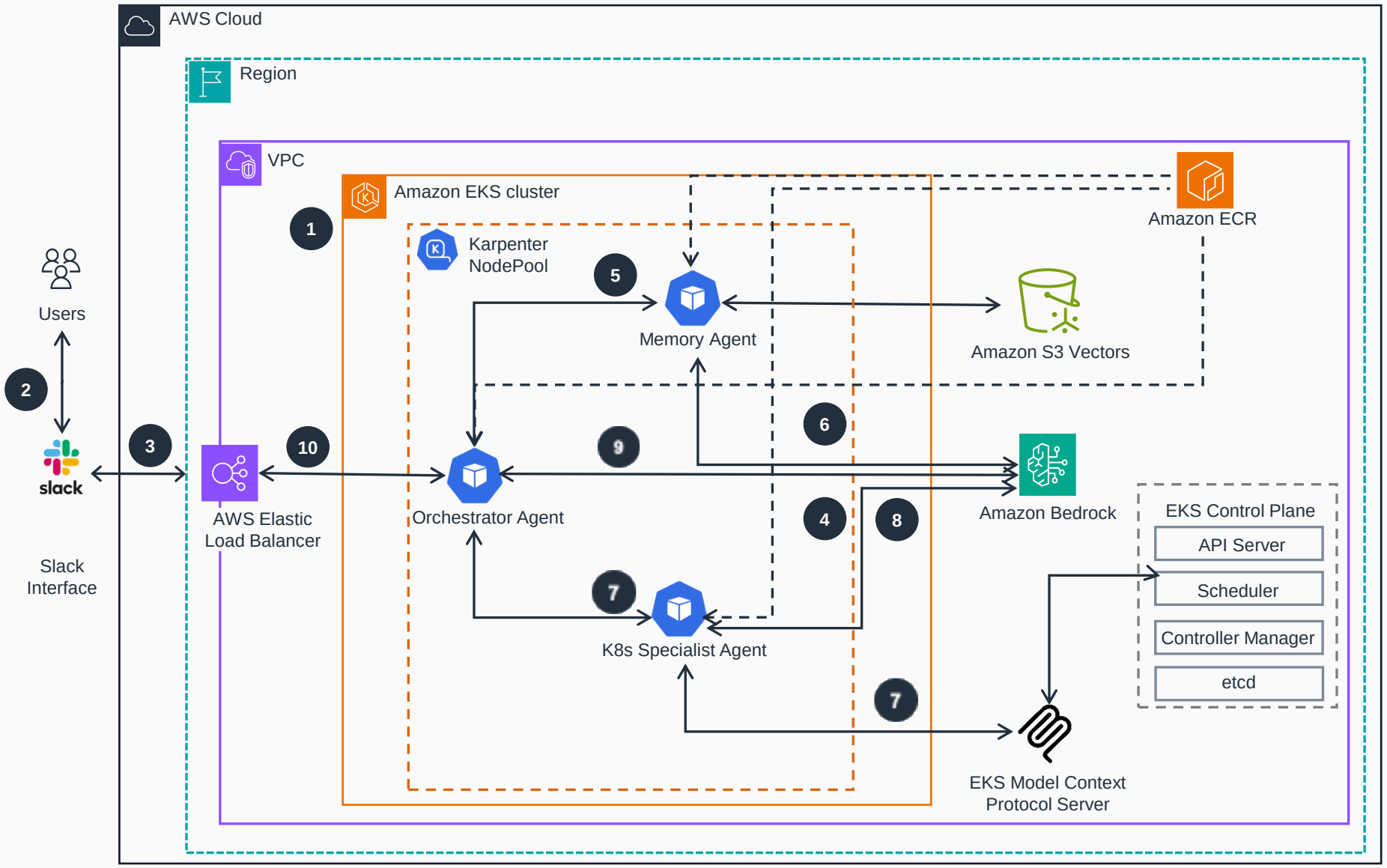
Reviewed for technical accuracy Jan,12 2026

AWS Reference Architecture

Guidance for Troubleshooting of Amazon EKS using Agentic AI workflow on AWS

Agentic AI Workflow

This architecture diagram shows troubleshooting Agentic AI workflow working with real-time EKS cluster data and integrated with AWS AI services and Slack



7 Orchestrator agent invokes the K8s Specialist agent, which utilizes the fully managed **Amazon EKS Model Context Protocol (MCP) Server** to execute read-only commands against the **Amazon EKS** API Server. The MCP Server authenticates with the Kubernetes API using IAM credentials, then gathers real-time cluster state (including pod status, deployments, services), retrieves relevant pod logs, collects recent events, and captures resource metrics (CPU, memory, network). This comprehensive data is structured and normalized by the MCP Server and returned to the K8s Specialist agent to provide context for the current problem analysis.

8 K8s Specialist agent sends the collected cluster data to **Anthropic Claude** model via **Amazon Bedrock** for intelligent issue analysis and resolution generation

9 Orchestrator agent synthesizes the historical context received from Memory agent and current cluster state from K8s Specialist, then uses **Anthropic Claude** model via **Amazon Bedrock** to generate comprehensive troubleshooting recommendations, which are stored in **Amazon S3** Vectors for future reference.

10 Orchestrator agent generates troubleshooting recommendations and sends them back to Users via integrated dedicated Slack channel. This illustrates troubleshooting using an increasingly popular “ChatOps” Platform Engineering pattern.

Reviewed for technical accuracy Jan,12 2026

AWS Reference Architecture