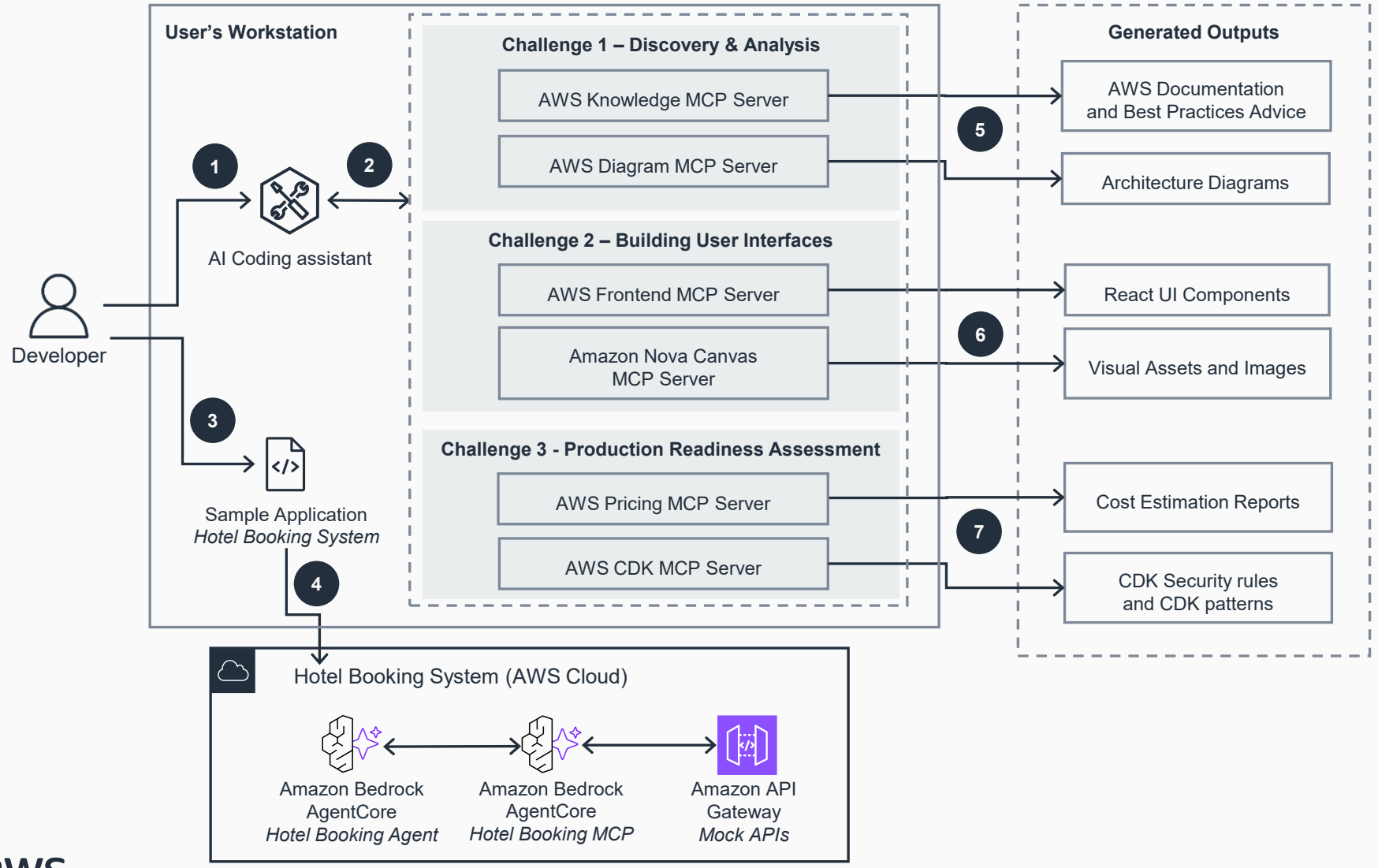
Guidance for Vibe Coding with AWS MCP Servers

Overview

This architecture diagram illustrates how to effectively develop AWS applications using AI assistants enhanced with AWS MCP Servers, demonstrated through a sample hotel booking application built on Amazon Bedrock AgentCore.



1. Developers install an AI coding assistant that supports the Model Context Protocol (MCP), such as Amazon Q Developer, Kiro, or compatible alternatives. MCP enables AI assistants to access structured, domain-specific capabilities through standardized server integrations.
2. Developers configure AWS MCP Servers in their development environment, enabling AI assistants to access AWS-specific capabilities. This guidance showcases six essential servers, with additional specialized servers available in the complete AWS MCP collection.
3. This guidance includes a comprehensive, real-world hotel booking application. Developers can deploy this reference implementation to their AWS account using the provided AWS CDK infrastructure, creating a working example for exploring vibe coding techniques with AWS MCP Servers.
4. The reference implementation runs on Amazon Bedrock AgentCore, where both the hotel booking agent and custom MCP server operate within the **Amazon Bedrock AgentCore** runtime, integrating with mock APIs for property resolution, reservations, and content moderation. See next slide for detailed architecture.
5. When exploring AWS services and architectures, developers leverage their AI assistant's integration with AWS Knowledge MCP Server to access official documentation and best practices. AWS Diagram MCP Server generates architecture visualizations, accelerating understanding of complex distributed systems.
6. Developers accelerate frontend development using AWS Frontend MCP Server to generate React components with AWS integration, while Nova Canvas MCP Server creates custom graphics and visual elements.
7. Production readiness assessment leverages AWS Pricing MCP Server for cost analysis and AWS CDK MCP Server for security evaluation through CDK Nag rules and AWS Solutions Constructs patterns, enabling data-driven deployment decisions.

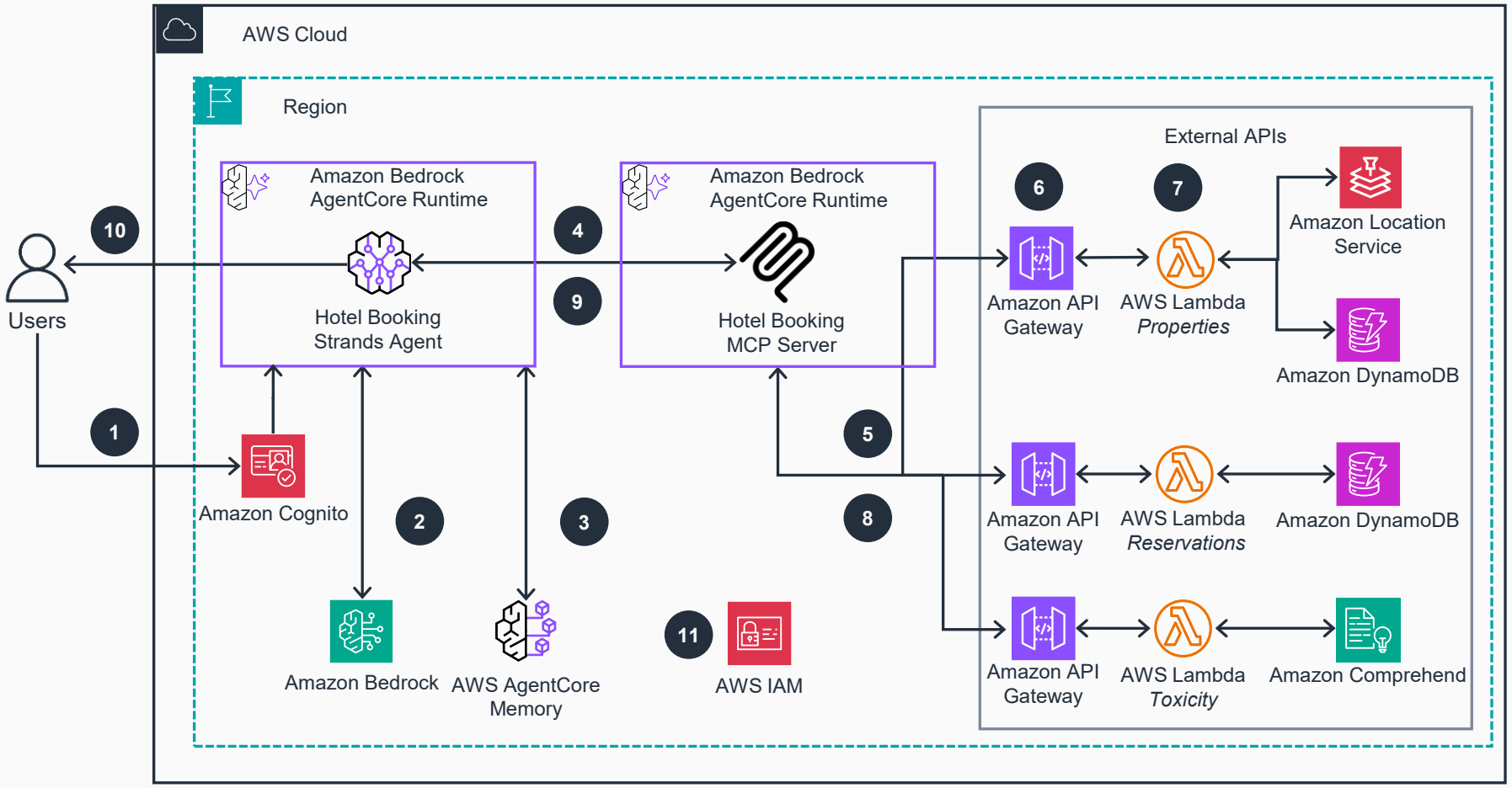
Reviewed for technical accuracy November 13, 2025

AWS Reference Architecture

Guidance for Vibe Coding with AWS MCP Servers

Hotel Booking System – Sample Application

This complete hotel booking system is provided as a realistic, hands-on example of Amazon Bedrock AgentCore in action. It demonstrates how AI agents and custom MCP servers orchestrate complex AWS services through natural language interactions.This slide shows steps 1-6.



1. A user, authenticated by **Amazon Cognito**, submits a request (e.g., "Find me a hotel in Seattle for next weekend") by invoking the Hotel Booking Agent. The agent is deployed on **Amazon Bedrock AgentCore. Amazon Bedrock AgentCore** Runtime provides a secure, serverless and purpose-built hosting environment for deploying and running AI agents or tools.

2. The agent, built using the **Strands Agents SDK**, invokes an **Amazon Bedrock** model to leverage LLM capabilities for natural language understanding.

3. The agent retrieves historical data about previous interactions with a user. **Amazon Bedrock AgentCore** Memory manages conversation context for the agent.

4. The agent connects to the Hotel Booking MCP Server, which is also deployed on **Amazon Bedrock AgentCore**, to discover and invoke tools required to complete the user's request.

5. Once a tool is selected, the agent calls it through its MCP Server. The MCP Server routes requests to **Amazon API Gateway.**

6. **Amazon API Gateway** exposes 3 different APIs that represent the tools of the MCP Server: Property Resolution, Reservations, and Toxicity Detection via corresponding **AWS Lambda** functions.

7. **AWS Lambda functions process the requests:**
   Property Resolution Lambda: Performs fuzzy matching against hotel records and uses **Amazon Location Service** to search properties not in **Amazon DynamoDB**. It returns top 5 matching hotels with details like location, amenities, and pricing.
   Reservations Lambda: Executes CRUD operations on reservation data, validates booking parameters (dates, guest count, room availability), generates confirmation numbers, and manages reservation status transitions (Booked → Confirmed → Cancelled).
   Toxicity Detection Lambda: Analyzes user input text using **Amazon Comprehend** for inappropriate content, applies allow list filtering, scores toxicity levels, and returns safety assessment results.

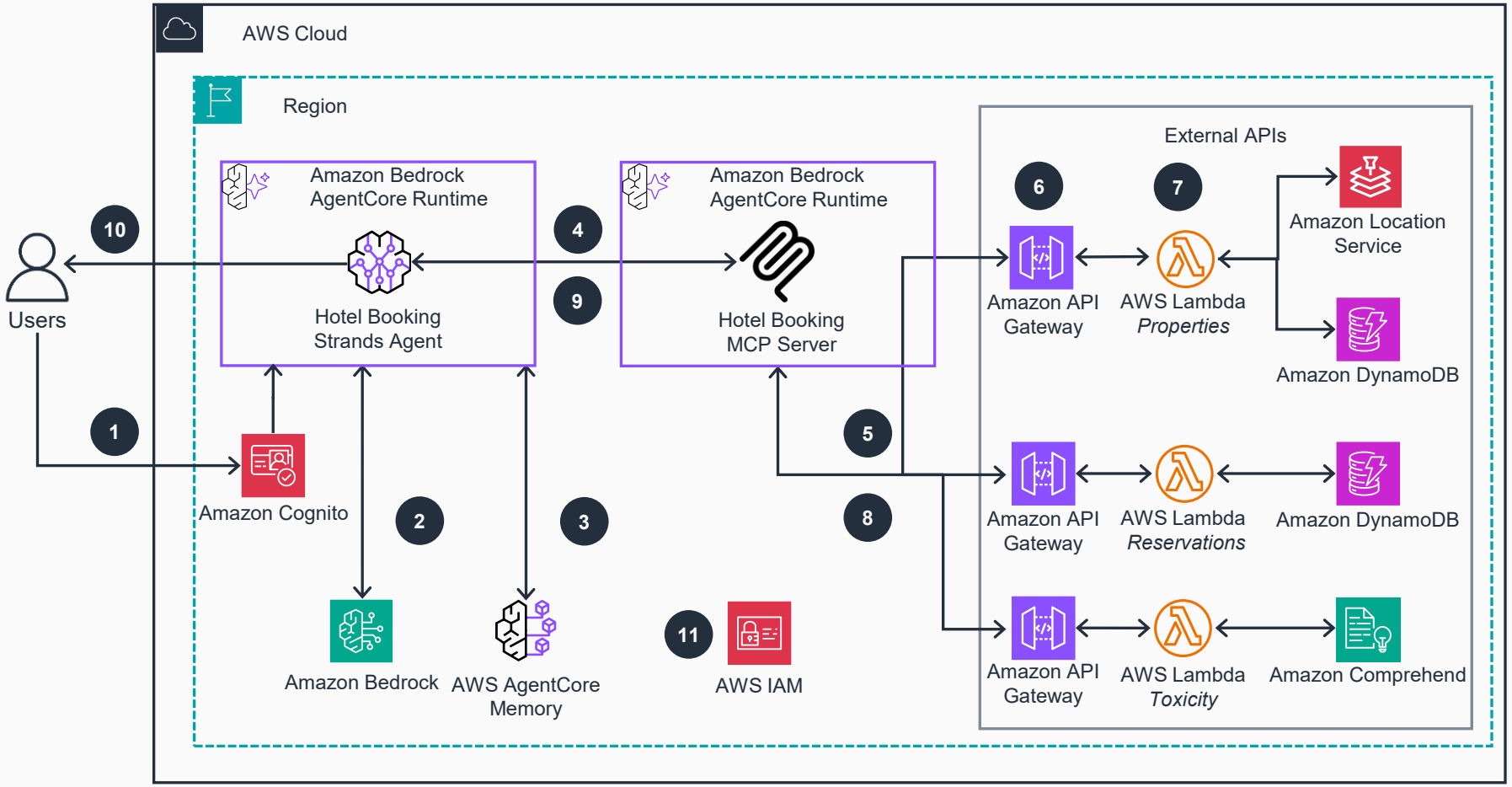
Reviewed for technical accuracy November 13, 2025

AWS Reference Architecture

Guidance for Vibe Coding with AWS MCP Servers

Hotel Booking System – Sample Application

This complete hotel booking system is provided as a realistic, hands-on example of Amazon Bedrock AgentCore in action. It demonstrates how AI agents and custom MCP servers orchestrate complex AWS services through natural language interactions.



8 API Responses are passed back to the MCP Server, so that the server can process them as tool results. The MCP Server transforms API responses into structured tool results, handles error conditions, formats data for agent consumption, and maintains request/response correlation for multi-step booking workflows.

9 The MCP Server tool results are sent back to the Hotel Booking agent for processing. The agent determines next steps, leveraging **Amazon Bedrock** foundation models to interpret the tool results, and decide the best course of action.

10 The final response (search results, booking confirmation, or error messages) are delivered back as a response to the User.

11 **AWS Identity and Access Management (AWS IAM)** secures the system with dedicated execution roles for the hotel booking agent, MCP server, and **AWS Lambda** functions. Each role implements least-privilege access to required AWS services including **Amazon Bedrock** models, **Amazon Bedrock AgentCore** Memory, and API resources.

Reviewed for technical accuracy November 13, 2025

AWS Reference Architecture