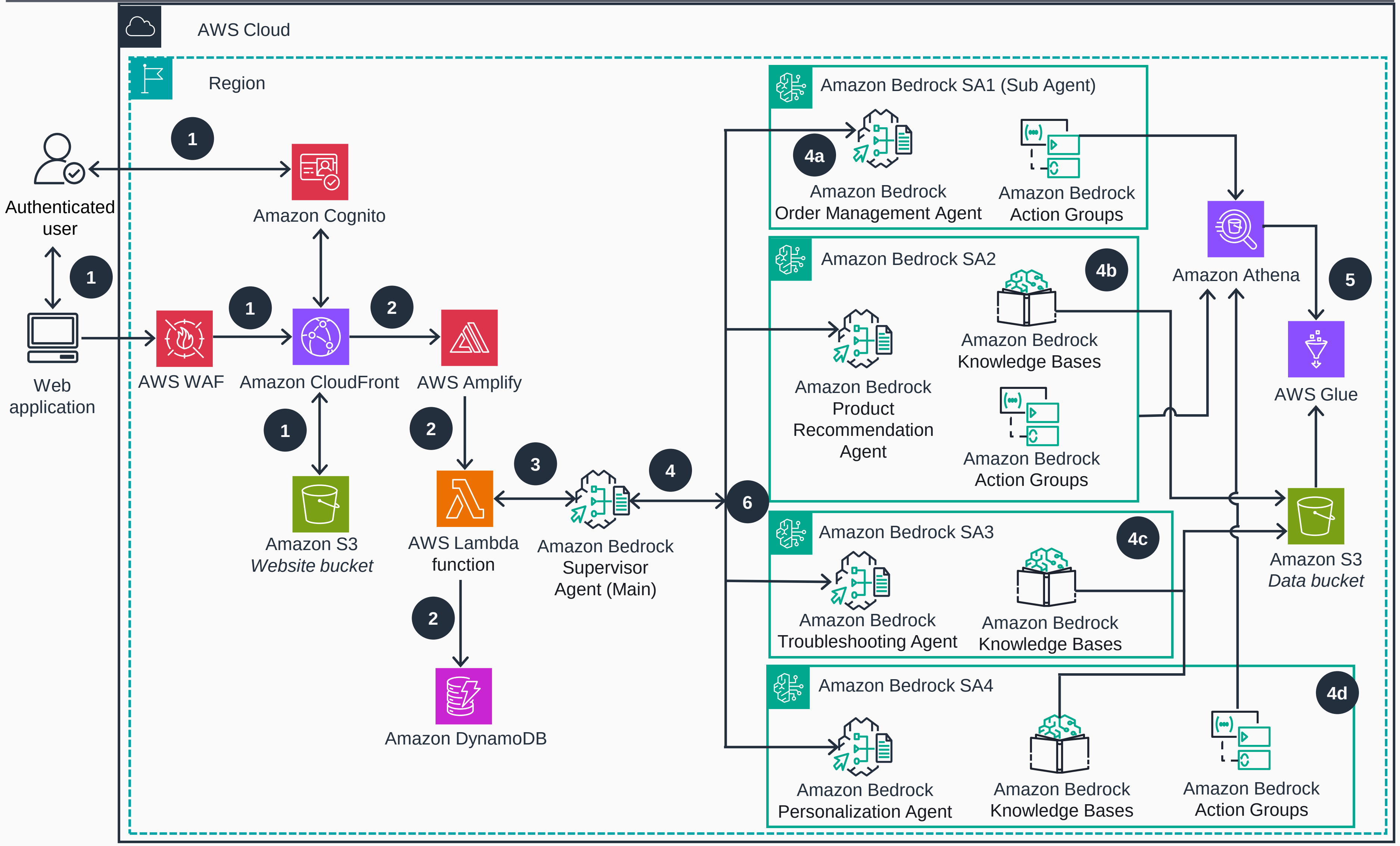
Guidance for Multi-Agent Orchestration on AWS

Using Amazon Bedrock multi-agent collaboration feature

This architecture showcases how a supervisor agent orchestrates multiple specialized sub-agents through Amazon Bedrock's native collaboration feature for comprehensive business scenarios. This pattern automatically handles task delegation and response aggregation across various functional agents with enterprise-grade reliability and built-in monitoring. This slide shows Steps 1-4c.



AWS Cloud
Region
1
Authenticated user
Amazon Cognito
1
Web application
AWS WAF
1
Amazon CloudFront
2
AWS Amplify
1
Amazon S3
Website bucket
2
AWS Lambda function
3
Amazon Bedrock Supervisor Agent (Main)
4
2
Amazon DynamoDB
6
Amazon Bedrock SA1 (Sub Agent)
4a
Amazon Bedrock Order Management Agent
Amazon Bedrock Action Groups
Amazon Bedrock SA2
4b
Amazon Bedrock Product Recommendation Agent
Amazon Bedrock Knowledge Bases
Amazon Bedrock Action Groups
Amazon Bedrock SA3
4c
Amazon Bedrock Troubleshooting Agent
Amazon Bedrock Knowledge Bases
Amazon Bedrock SA4
4d
Amazon Bedrock Personalization Agent
Amazon Bedrock Knowledge Bases
Amazon Bedrock Action Groups
Amazon Athena
5
AWS Glue
Amazon S3
Data bucket

1. The user accesses the web application through **AWS WAF** and **Amazon CloudFront** which delivers content from the **Amazon Simple Storage Service (Amazon S3)** website bucket. **Amazon Cognito** handles authentication.

2. After authentication, user requests are sent to **AWS Amplify**, which serves as the entry point for all interactions. **Amplify** validates the request and routes it to the appropriate **AWS Lambda** function for processing, maintaining a secure and scalable communication channel. **Amazon DynamoDB** stores session data.

3. The **Lambda** function processes the incoming request and communicates with the **Amazon Bedrock** Supervisor Agent (Main).

4. The **Amazon Bedrock** Supervisor Agent (Main) analyzes the user query to determine intent and routes it to the appropriate sub agent. This central orchestrator maintains context across the conversation and ensures requests are handled by the most suitable sub agent.

4a For order-related queries, the Order Management Agent retrieves data from the order management database in **Amazon Athena**, accessing orders and inventory tables through its action groups. The action groups execute SQL queries and format structured responses about order status, shipping details, and inventory availability.

4b When product recommendations are needed, this specialized agent accesses the product recommendation database in **Athena** while its knowledge base provides unstructured customer feedback data from **Amazon S3**. Action groups perform recommendation algorithms and format product suggestions with relevant details.

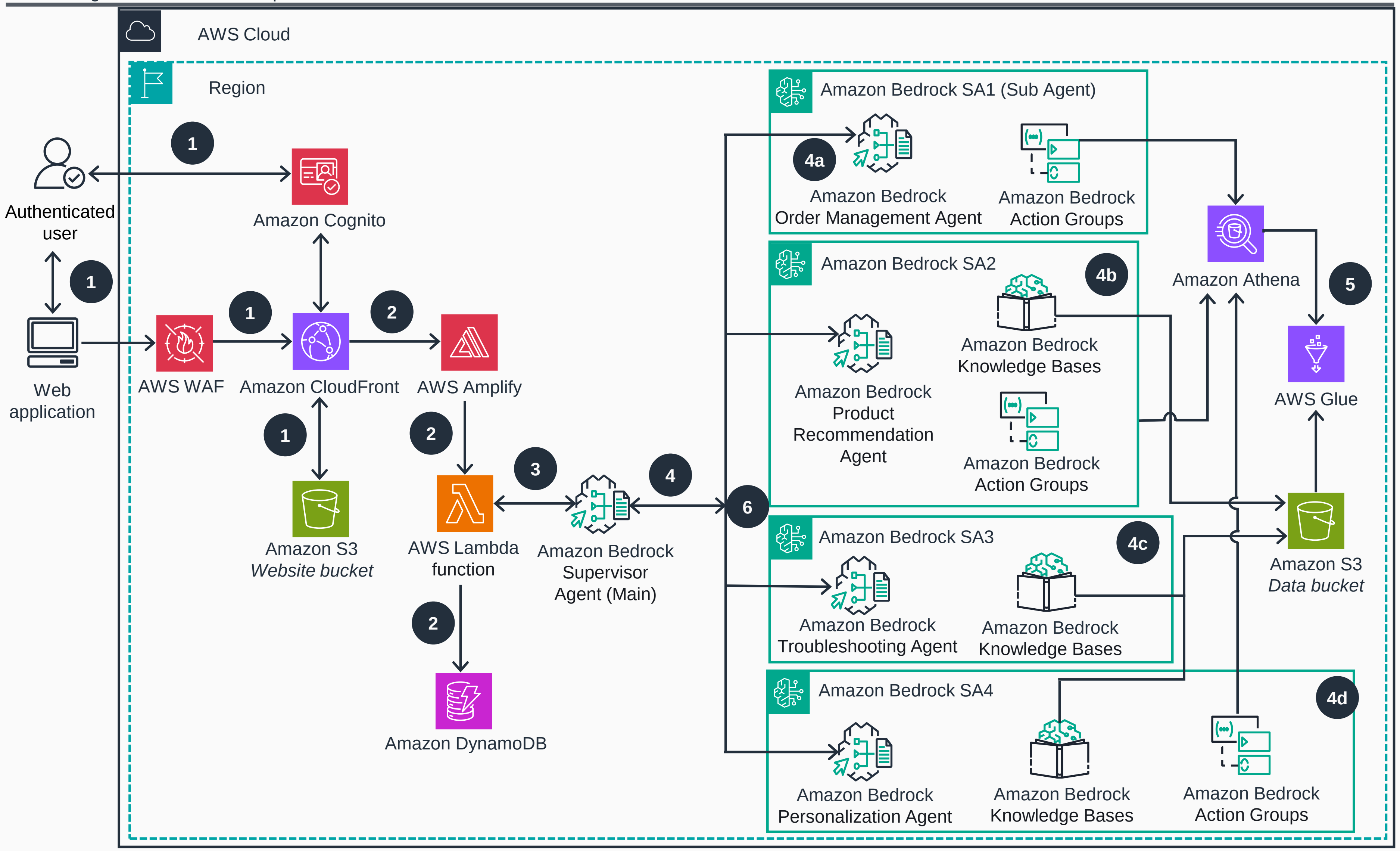
4c For technical issues, the Troubleshooting Agent accesses its knowledge base containing FAQs and Troubleshooting Guide document collections. It uses vector search capabilities to match customer issues with relevant troubleshooting content and to retrieve step-by-step solutions without requiring action groups.

AWS Reference Architecture

Guidance for Multi-Agent Orchestration on AWS

Using Amazon Bedrock multi-agent collaboration feature

This architecture showcases how a supervisor agent orchestrates multiple specialized sub-agents through Amazon Bedrock's native collaboration feature for comprehensive business scenarios. This pattern automatically handles task delegation and response aggregation across various functional agents with enterprise-grade reliability and built-in monitoring. This slide shows Steps 4d-6.



AWS Cloud
Region
Authenticated user
Web application
Amazon Cognito
AWS WAF
Amazon CloudFront
AWS Amplify
Amazon S3
Website bucket
AWS Lambda function
Amazon Bedrock Supervisor Agent (Main)
Amazon DynamoDB
1
2
3
4
4a
4b
4c
4d
5
6
Amazon Bedrock SA1 (Sub Agent)
Amazon Bedrock Order Management Agent
Amazon Bedrock Action Groups
Amazon Bedrock SA2
Amazon Bedrock Product Recommendation Agent
Amazon Bedrock Knowledge Bases
Amazon Bedrock Action Groups
Amazon Bedrock SA3
Amazon Bedrock Troubleshooting Agent
Amazon Bedrock Knowledge Bases
Amazon Bedrock SA4
Amazon Bedrock Personalization Agent
Amazon Bedrock Knowledge Bases
Amazon Bedrock Action Groups
Amazon Athena
AWS Glue
Amazon S3
Data bucket

4d For personalization needs, the Personalization Agent accesses the personalization database in **Athena**, querying the customer's preferences table through action groups. These action groups execute tailored SQL queries, perform preference analysis, and format responses. Its knowledge base contains browser history data from **Amazon S3** that reveals actual customer behavior patterns, complementing the structured data to create a comprehensive view of individual customer profiles and past interactions.

5 Sub agents construct and execute SQL queries against **Athena**, which uses the **AWS Glue Data Catalog** to understand the schema and location of data, then queries the data directly in **Amazon S3** without requiring data movement or transformation.

6 After gathering necessary information from databases and knowledge bases, the sub agents generate a comprehensive response. This response is then sent back through the Supervisor Agent to the **Lambda** function, then **Amplify**, and finally to the user's web interface.

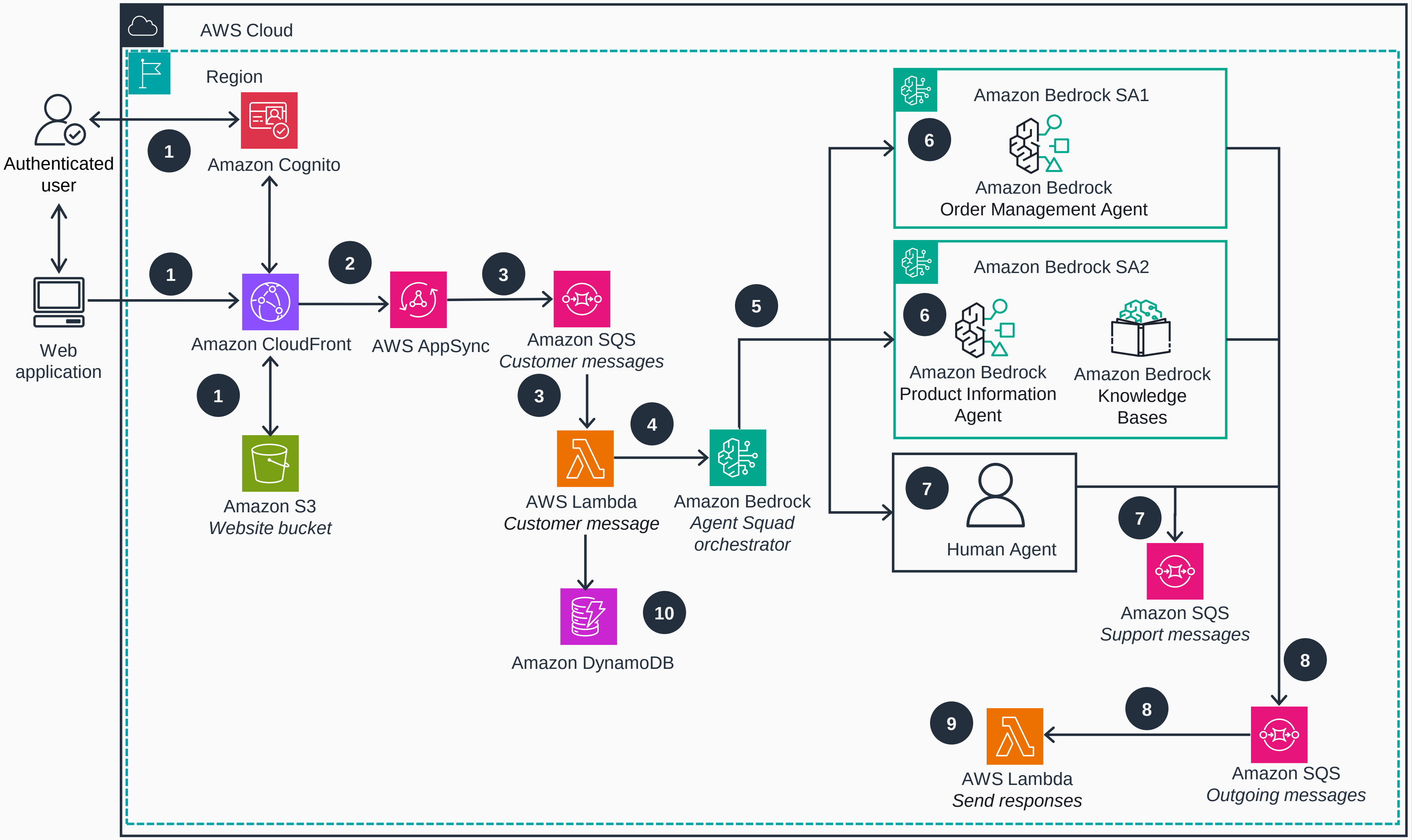
Reviewed for technical accuracy September 11, 2025

AWS Reference Architecture

Guidance for Multi-Agent Orchestration on AWS

Using Agent Squad

This architecture diagram shows how specialized AI agents operate as independent microservices, each handling specific business domains through custom coordination logic. This pattern enables complete control over agent behavior and seamless integration with both external systems and human agents through message queues. This slide shows Steps 1-8.



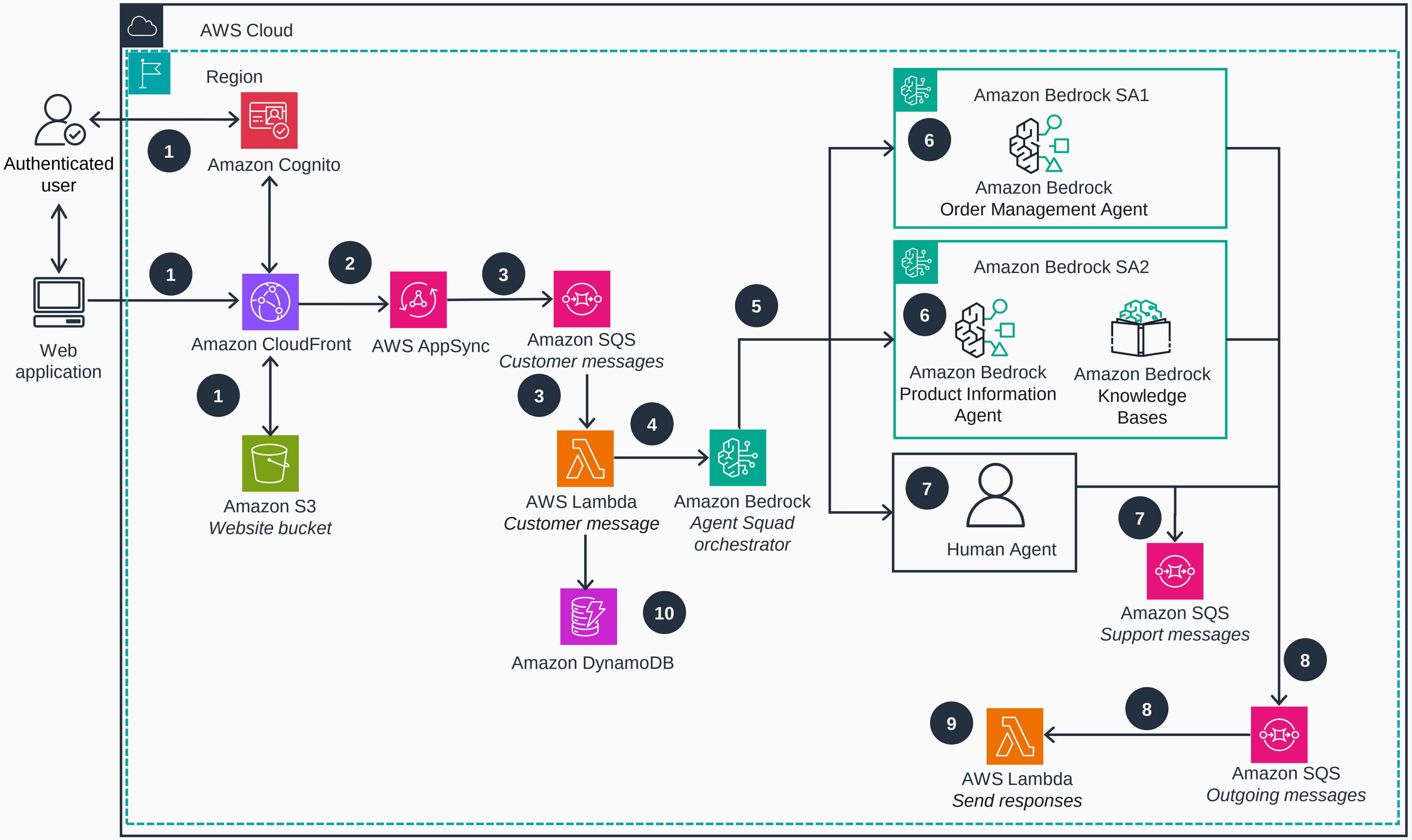
1. The user accesses the web application through **Amazon CloudFront**, which delivers content from the **Amazon S3** Website bucket, while **Amazon Cognito** handles authentication and provides temporary credentials via Identity Pool.
2. The authenticated user sends messages through the web UI, which communicates with **AWS AppSync** GraphQL API to process queries, mutations, and subscriptions for real-time communication.
3. **AWS AppSync** routes customer messages to the **Amazon SQS** Customer Messages Queue, which triggers the **AWS Lambda** Customer Message Handler function to process incoming customer inquiries.
4. **AWS Lambda** Customer Messages initializes the **Agent Squad orchestrator**, which uses **Amazon Bedrock** Classifier to analyze the message content and determine which specialized AI agent should handle the request.
5. Based on the classification, the message is routed to one of three agents: the Order Management Agent (using Claude 3 Sonnet), the Product Information Agent (using Claude 3 Haiku), or the Human Agent for complex cases requiring human intervention.
6. The Order Management Agent handles order-related inquiries using tools for order lookup, shipment tracking, and return processing, while the Product Information Agent retrieves product details from the **Amazon Bedrock** Knowledge Base connected to OpenSearch Serverless.
7. When complex inquiries are routed to the Human Agent, it sends the customer message to the **Amazon SQS** Support Messages Queue and notifies the customer that their request will be handled by a human support representative.
8. Agent responses are sent to the **Amazon SQS** Outgoing Messages Queue, which triggers the **AWS Lambda** Send Response Handler to process and format the responses.

AWS Reference Architecture

Guidance for Multi-Agent Orchestration on AWS

Using Agent Squad

This architecture diagram shows how specialized AI agents operate as independent microservices, each handling specific business domains through custom coordination logic. This pattern enables complete control over agent behavior and seamless integration with both external systems and human agents through message queues. This slide shows Steps 9-10.



9 The Send Response Lambda sends the formatted responses back to **AWS AppSync**, which delivers them to the web UI through GraphQL subscriptions for real-time updates to the customer interface.

10 Throughout this process, **Amazon DynamoDB** stores conversation history and session data, while **AWS IAM** roles and policies ensure secure access to all services and resources.

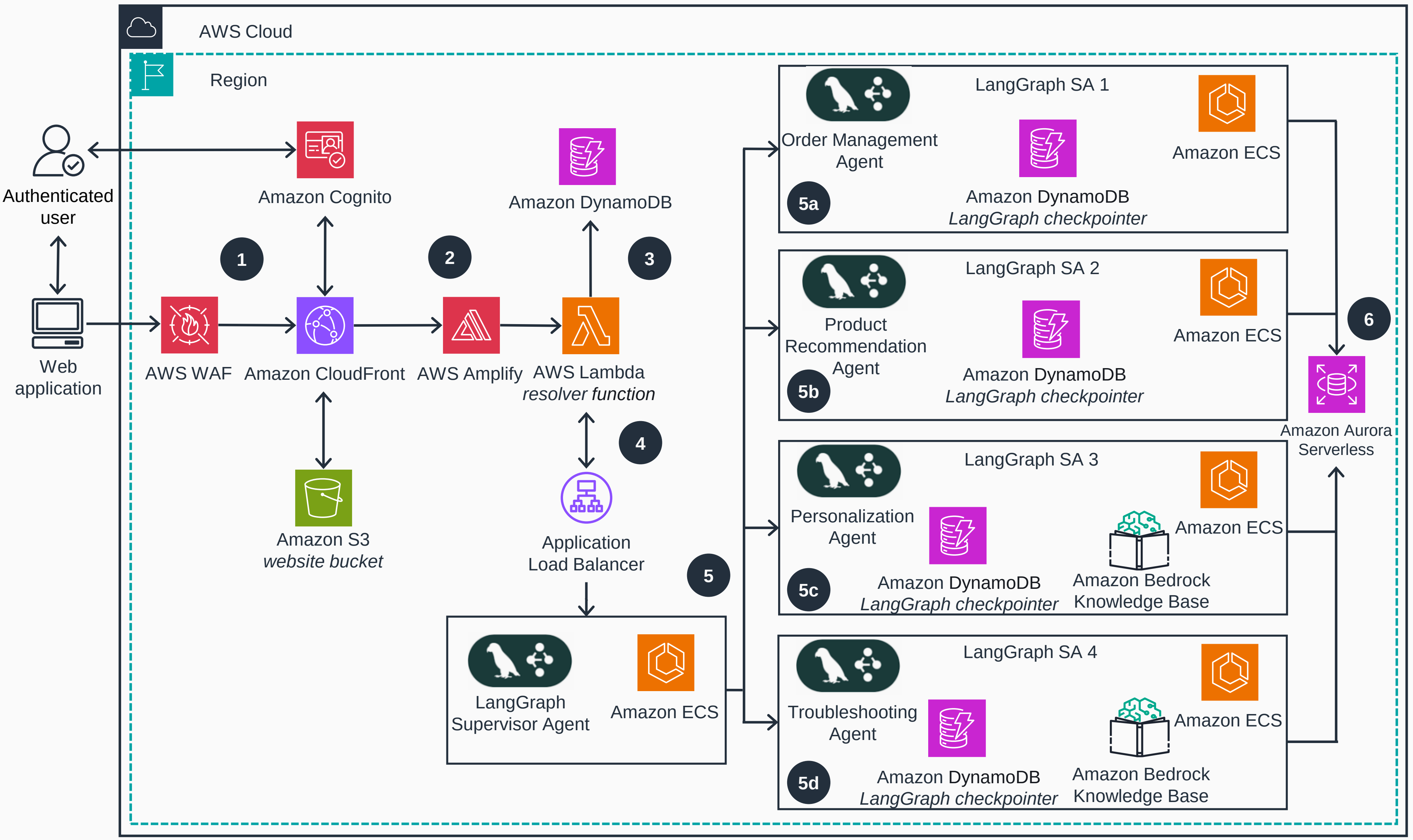
Reviewed for technical accuracy September 11, 2025

AWS Reference Architecture

Guidance for Multi-Agent Orchestration on AWS

LangGraph

This architecture diagram shows a LangGraph-powered supervisor agent running on Amazon ECS that intelligently coordinates four specialized sub-agents through LangGraph's agent orchestration framework, enabling seamless task delegation, context sharing, and response synthesis across distributed agents for comprehensive customer support scenarios. This slide shows Steps 1-5b.



1. The user accesses the web application through **AWS Web Application Firewall (AWS WAF)** and **Amazon CloudFront**, which delivers content from the **Amazon Simple Storage Service** (**Amazon S3**) website bucket. **Amazon Cognito** handles user authentication and authorization.

2. After authentication, **AWS Amplify** handles user requests, serving as the entry point for all interactions. **Amplify** validates the request and routes it to the **AWS Lambda** resolver function for processing. **Amazon DynamoDB** stores session data and user interaction history.

3. The **AWS Lambda** resolver function processes the incoming request and forwards it to the **Application Load Balancer**, which distributes the traffic for high availability and optimal performance.

4. The **Application Load Balancer** routes the request to the LangGraph Supervisor Agent running on **Amazon Elastic Container Service (Amazon ECS)**. This supervisor agent acts as the central orchestrator that analyzes user queries and determines the appropriate specialized agent to handle the request.

5. The LangGraph Supervisor Agent analyzes the user query to determine intent and routes it to the appropriate specialized sub-agent based on the request type:

5a. LangGraph Sub-Agent 1 (SA1) Order Management Agent running on **Amazon ECS** handles order-related queries. It uses an **Amazon DynamoDB**-based checkpoint saver implementation for LangGraph to maintain conversation context and retrieve order data, inventory information, and shipping details.

5b. LangGraph Sub-Agent 2 (SA2) Product Recommendation Agent running on **Amazon ECS** processes product recommendation requests. It accesses product databases through **Amazon DynamoDB**-based checkpoint saver implementation for LangGraph and uses machine learning algorithms to provide personalized product suggestions.

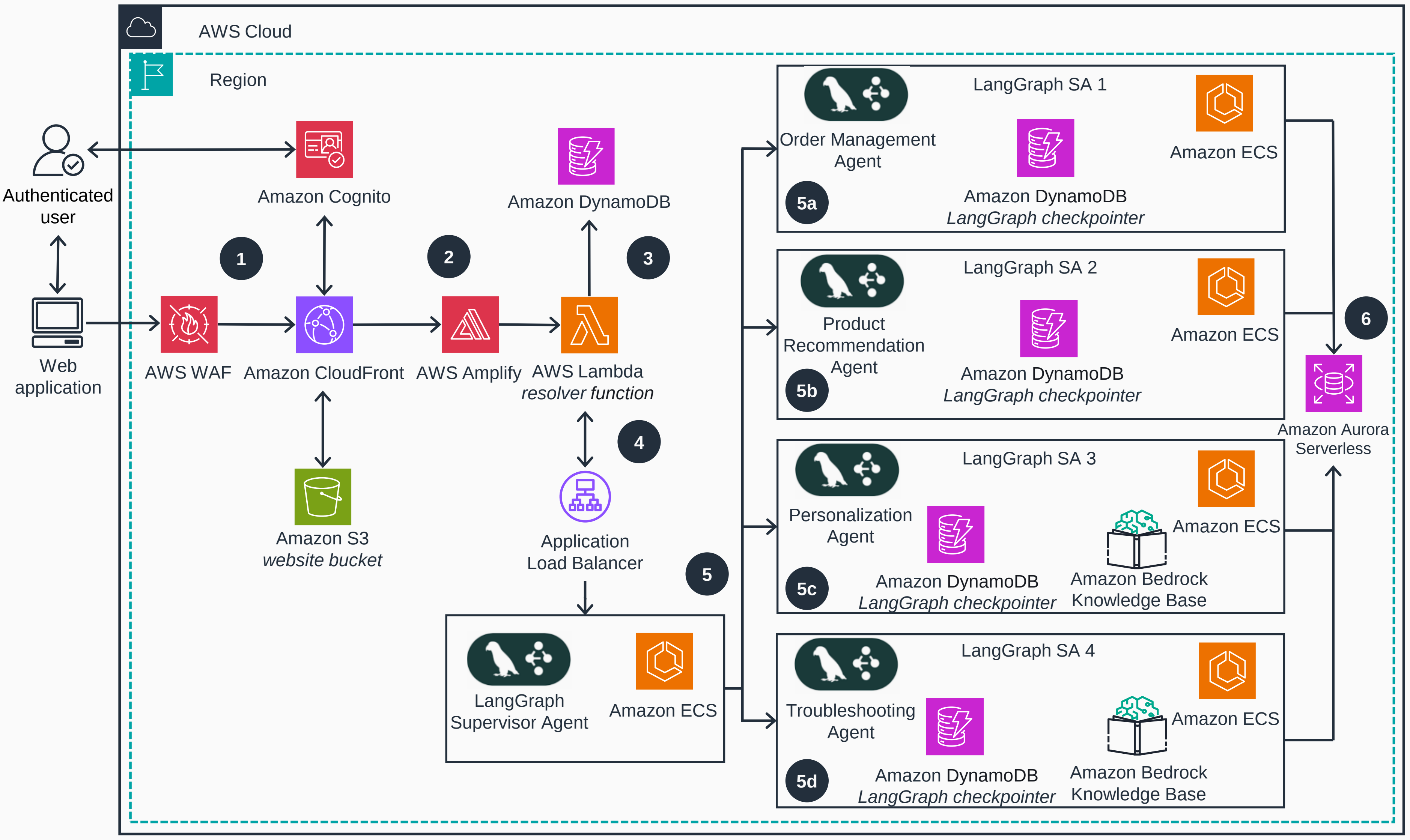
Reviewed for technical accuracy September 11, 2025

AWS Reference Architecture

Guidance for Multi-Agent Orchestration on AWS

LangGraph

This architecture diagram shows a LangGraph-powered supervisor agent running on Amazon ECS that intelligently coordinates four specialized sub-agents through LangGraph's agent orchestration framework, enabling seamless task delegation, context sharing, and response synthesis across distributed agents for comprehensive customer support scenarios. This slide shows Steps 5c-6.



5c LangGraph Sub-Agent 3 (SA3) Personalization Agent running on **Amazon ECS** manages user-specific customization and personalization requests. It connects to an **Amazon Bedrock Knowledge Base** to access customer preference data and behavioral patterns.

5d LangGraph Sub-Agent 4 (SA4) Troubleshooting Agent running on **Amazon ECS** handles technical support queries. It accesses the **Amazon Bedrock Knowledge Base** containing FAQs, troubleshooting guides, and technical documentation to provide step-by-step solutions.

6 **Amazon Aurora Serverless v2** provides persistent relational database storage for structured data including user profiles, order history, product catalogs, and system metadata that the agents can query as needed.

AWS Reference Architecture