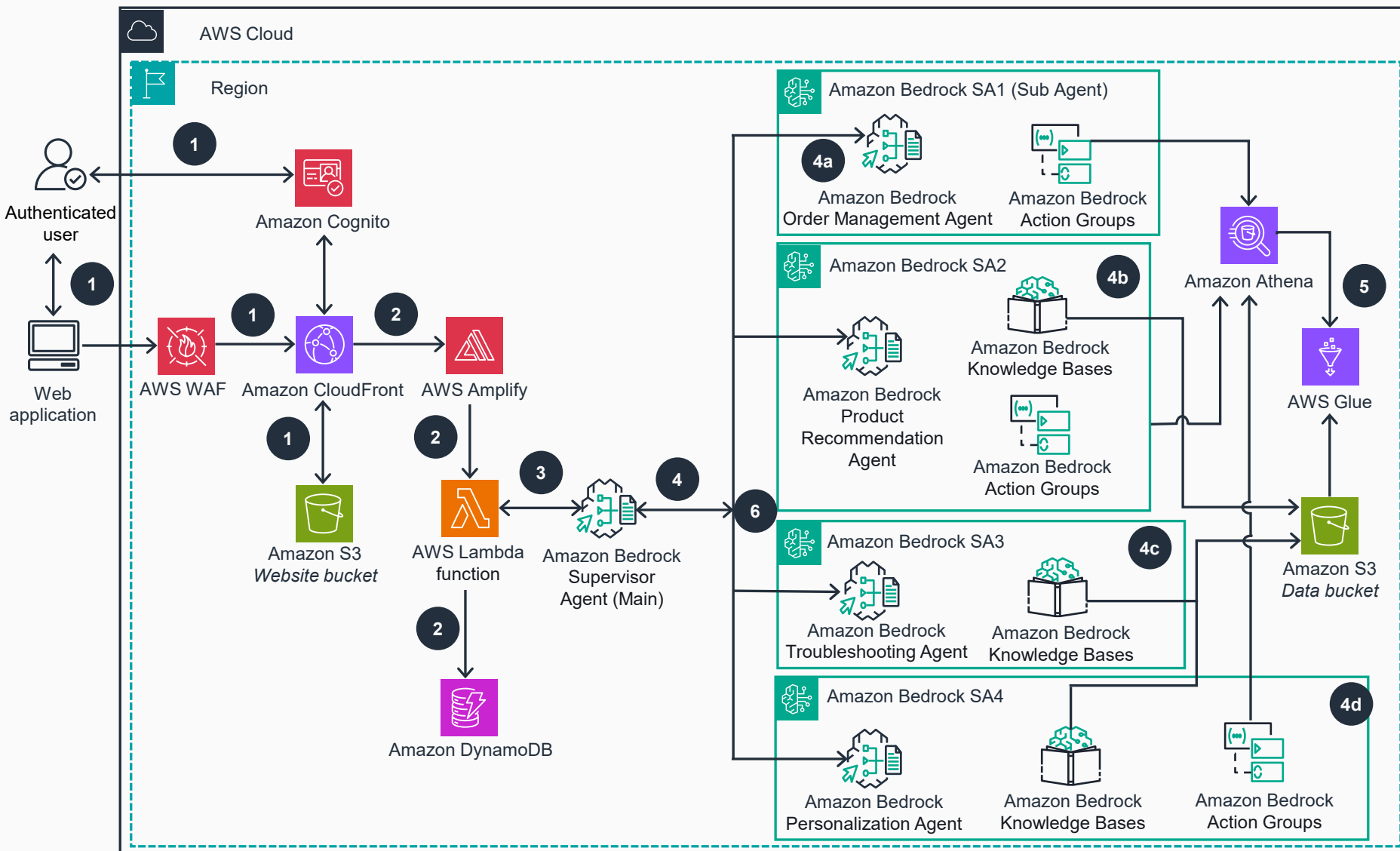# Guidance for Multi-Agent Orchestration on AWS

Amazon Bedrock Multi-Agent Collaboration

This architecture diagram shows how to build an intelligent customer support system using Amazon Bedrock multi-agent collaboration, where specialized agents work together to deliver data-driven customer support.


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

Amazon Bedrock Multi-Agent Collaboration

This architecture diagram shows how to build an intelligent customer support system using Amazon Bedrock multi-agent collaboration, where specialized agents work together to deliver data-driven customer support.
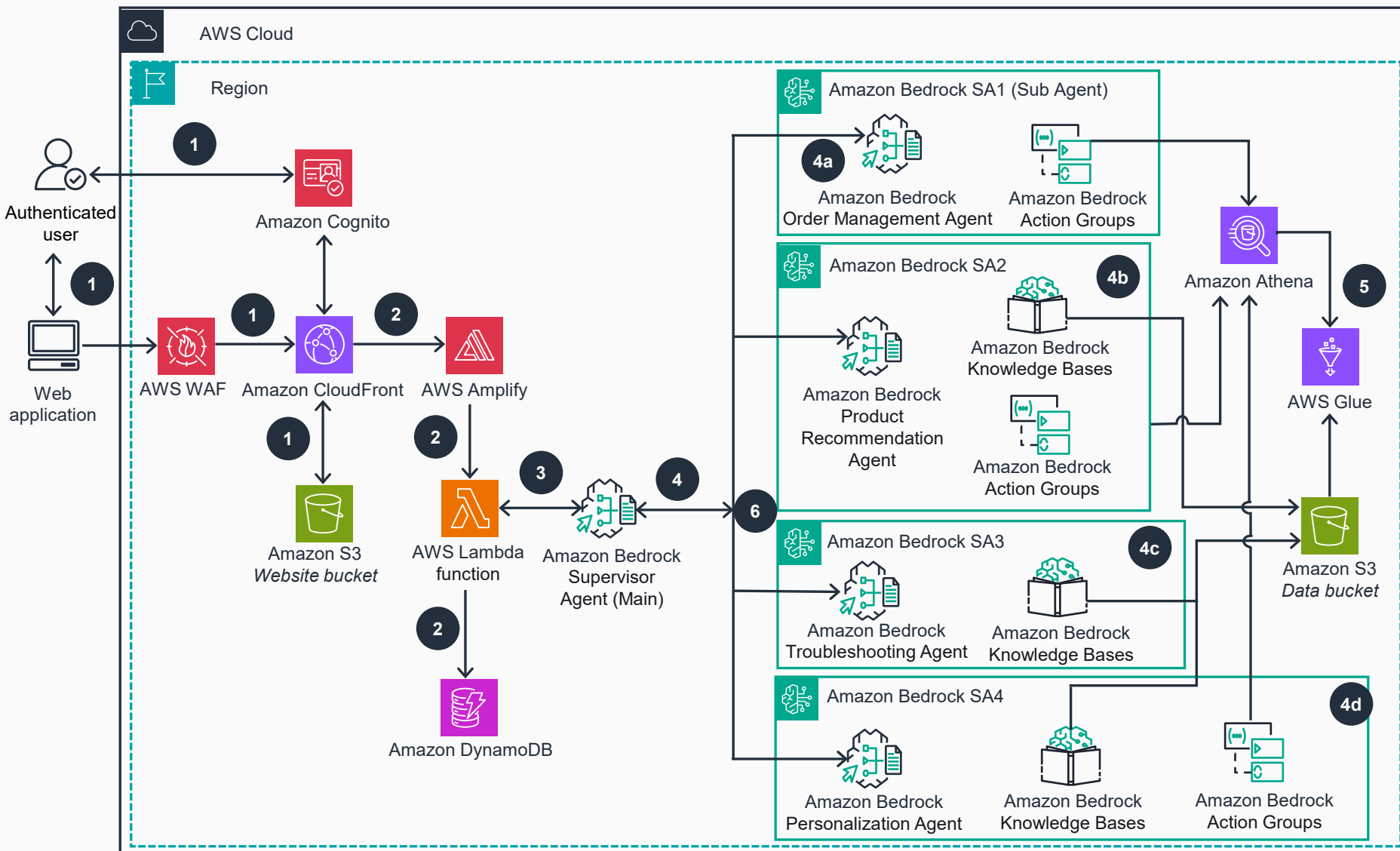
4d For personalization needs, the Personalization Agent accesses the personalization database in **Athena**, querying the customer's preferences table through action groups. These action groups execute tailored SQL queries, perform preference analysis, and format responses. Its knowledge base contains 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.
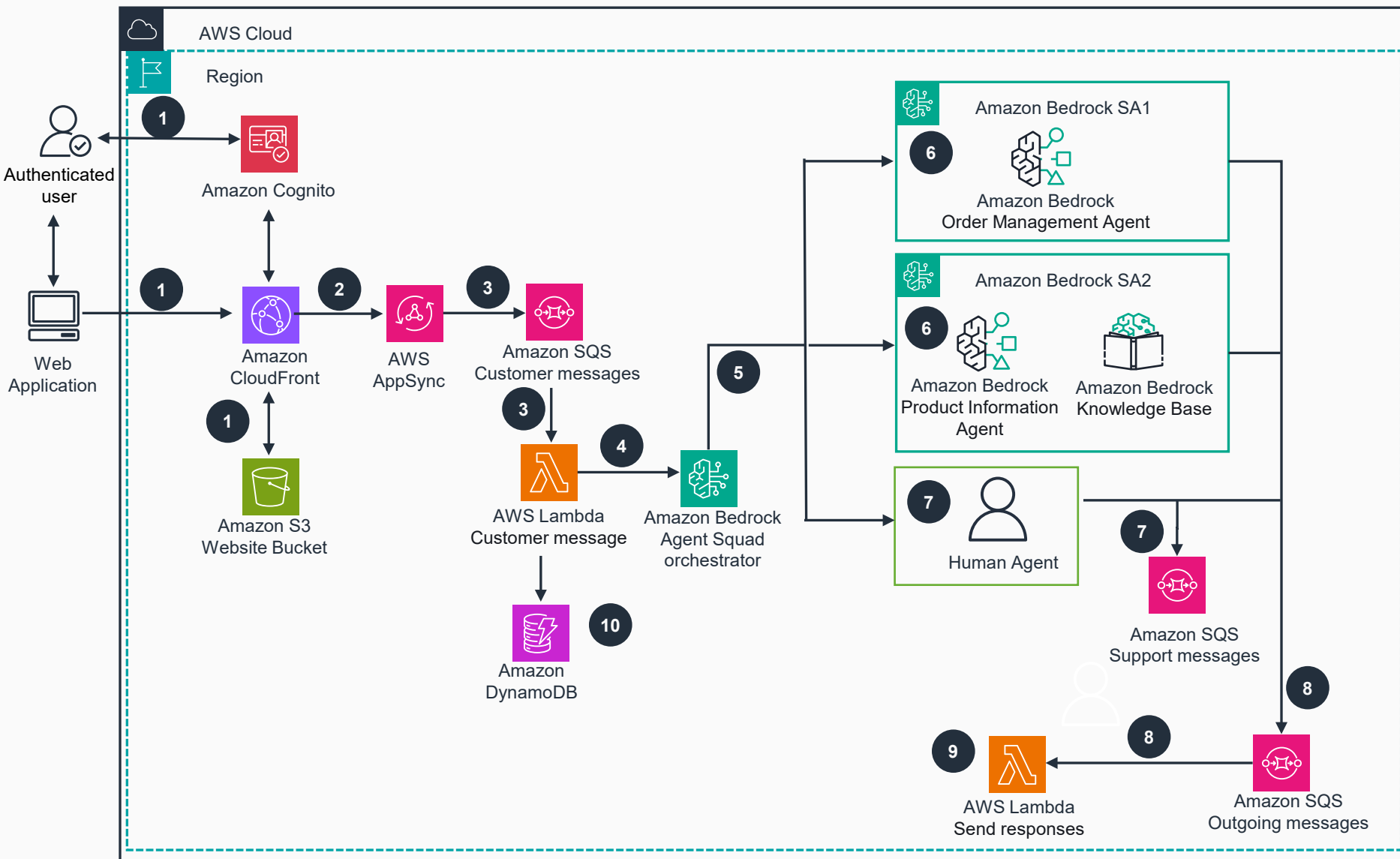
6 After gathering necessary information from databases and knowledge bases, the sub agents generate a comprehensive response. This response is then sent back through the Supervisor Agent to the **Lambda** function, then **Amplify**, and finally to the user's web interface.

AWS Reference Architecture

Guidance for Multi-Agent Orchestration on AWS

Agent Squad

This architecture diagram shows how to build an intelligent customer support system using Amazon Bedrock multi-agent collaboration, where specialized agents work together to deliver data-driven customer support.

AWS Cloud

Region

Authenticated user

Web Application

1 — Amazon Cognito

1 — Amazon CloudFront

1 — Amazon S3 Website Bucket

2 — AWS AppSync

3 — Amazon SQS Customer messages

3 — AWS Lambda Customer message

4 — Amazon Bedrock Agent Squad orchestrator

10 — Amazon DynamoDB

Amazon Bedrock SA1
6 — Amazon Bedrock Order Management Agent

Amazon Bedrock SA2
6 — Amazon Bedrock Product Information Agent
Amazon Bedrock Knowledge Base

7 — Human Agent

7 — Amazon SQS Support messages

8 — Amazon SQS Outgoing messages

9 — AWS Lambda Send responses

5

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.

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.

AWS Reference Architecture

Guidance for Multi-Agent Orchestration on AWS

LangGraph

This architecture showcases a LangGraph-powered supervisor agent running on Amazon ECS that intelligently coordinates four specialized sub-agents (Order Management, Product Recommendation, Personalization, and Troubleshooting) through LangGraph's agent orchestration framework, enabling seamless task delegation, context sharing, and response synthesis across distributed agents for comprehensive customer support scenarios.
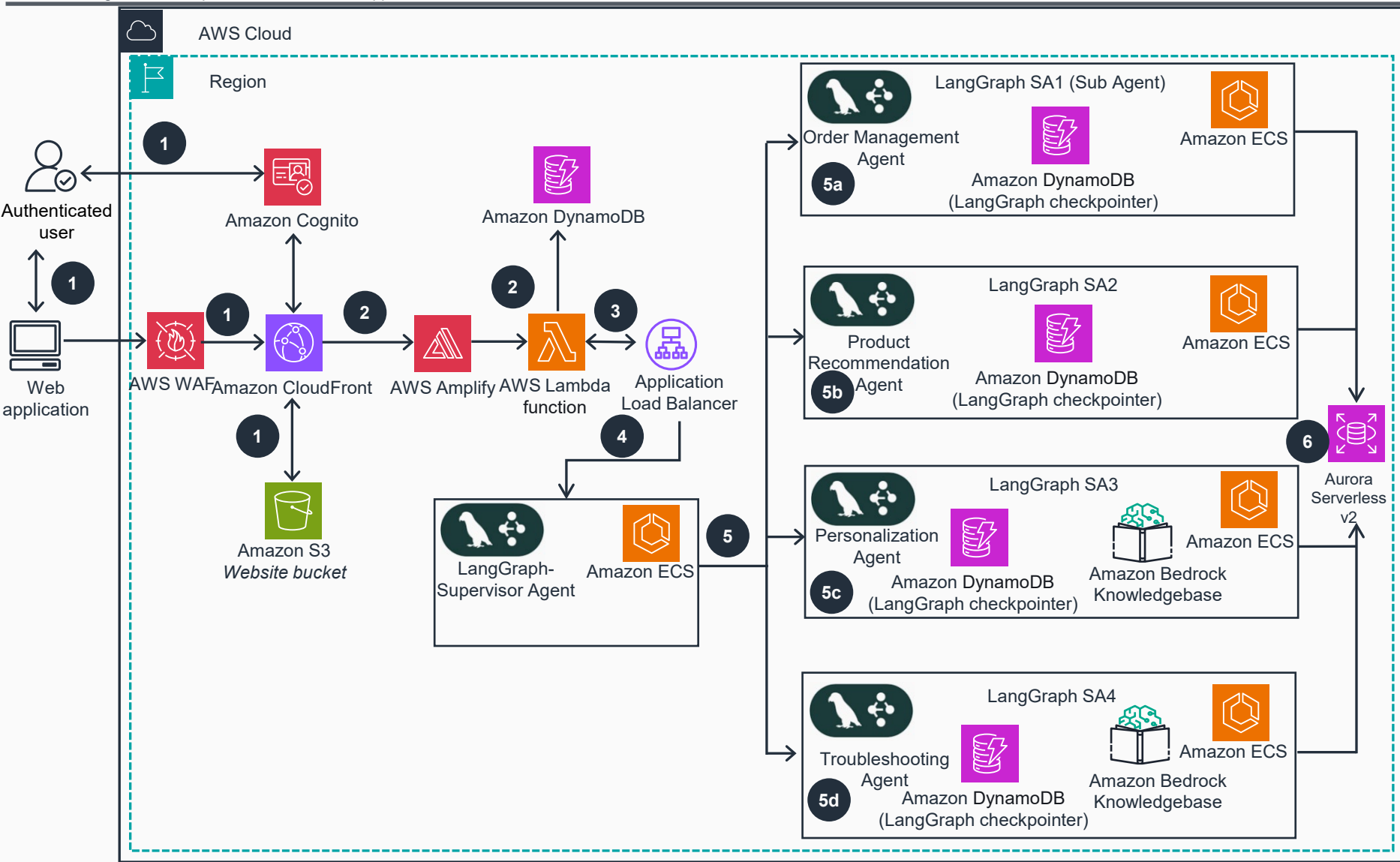
1. The user accesses the web application through **AWS WAF** and **Amazon CloudFront** which delivers content from the **Amazon Simple Storage Service** (**Amazon S3**) website bucket. **Amazon Cognito** handles user authentication and authorization.

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 **AWS Lambda** resolver function for processing. **Amazon DynamoDB** stores session data and user interaction history.

3. The **AWS Lambda** 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 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 SA1 (**Order Management Agent**) running on **Amazon ECS** handles order-related queries. It uses **Amazon DynamoDB** based checkpoint saver implementation for LangGraph to maintain conversation context and retrieves order data, inventory information, and shipping details.
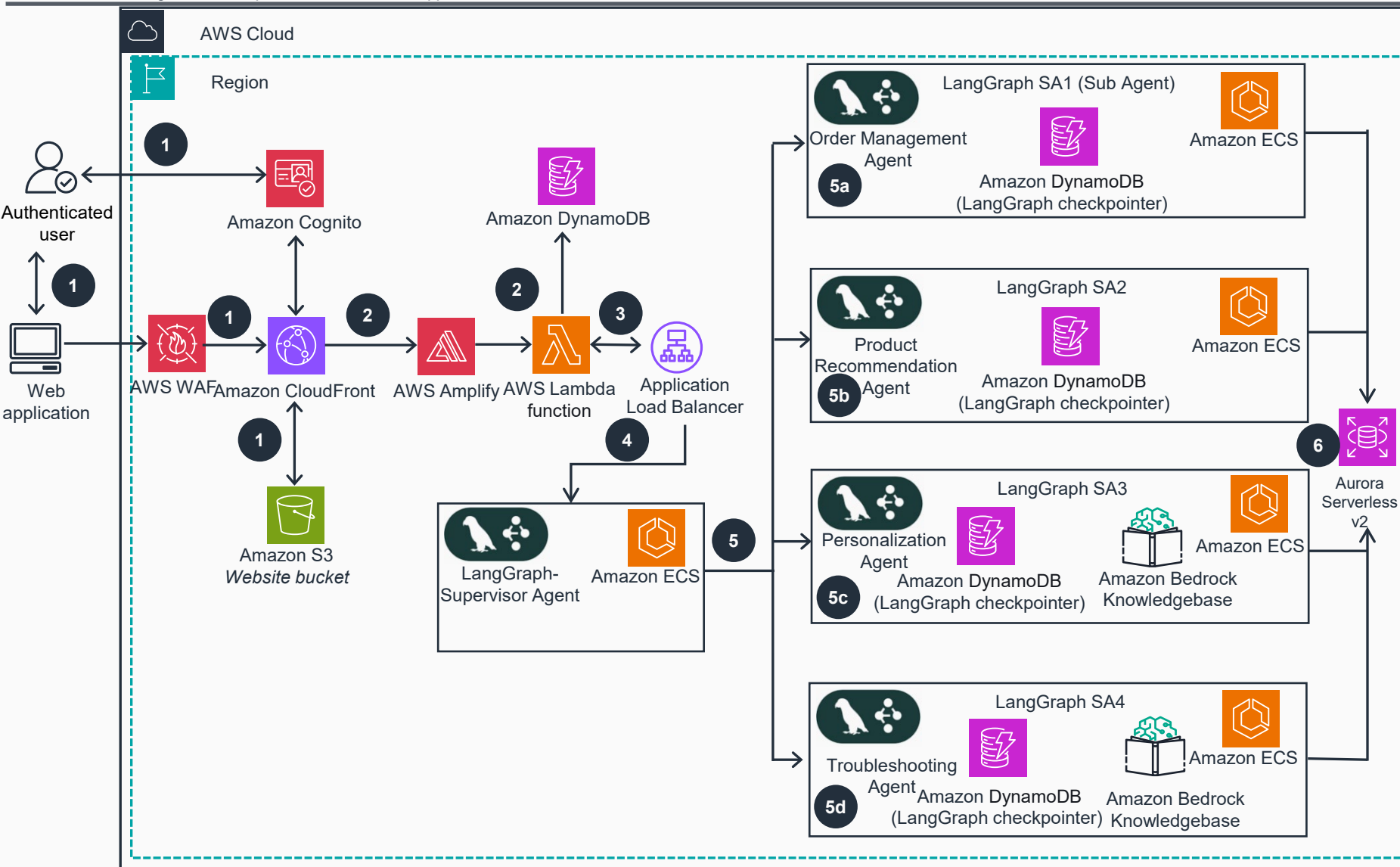
5b. LangGraph SA2 (**Product Recommendation Agent**) running on **Amazon ECS** processes product recommendation requests. It accesses product databases through **Amazon DynamoDB** based checkpoint saver implementation for LangGraph and uses machine learning algorithms to provide personalized product suggestions.

AWS Reference Architecture

Guidance for Multi-Agent Orchestration on AWS

LangGraph

This architecture showcases a LangGraph-powered supervisor agent running on Amazon ECS that intelligently coordinates four specialized sub-agents (Order Management, Product Recommendation, Personalization, and Troubleshooting) through LangGraph's multi-agent collaboration framework, enabling seamless task delegation, context sharing, and response synthesis across distributed agents for comprehensive customer support scenarios.

AWS Cloud

Region

Authenticated user

Web application

1 AWS WAF

1 Amazon CloudFront

2 AWS Amplify

2 AWS Lambda function

3 Application Load Balancer

Amazon Cognito

Amazon DynamoDB

1 Amazon S3 *Website bucket*

4 LangGraph-Supervisor Agent — Amazon ECS

5

5a LangGraph SA1 (Sub Agent) — Order Management Agent
Amazon DynamoDB (LangGraph checkpointer)
Amazon ECS

5b LangGraph SA2 — Product Recommendation Agent
Amazon DynamoDB (LangGraph checkpointer)
Amazon ECS

5c LangGraph SA3 — Personalization Agent
Amazon DynamoDB (LangGraph checkpointer)
Amazon Bedrock Knowledgebase
Amazon ECS

5d LangGraph SA4 — Troubleshooting Agent
Amazon DynamoDB (LangGraph checkpointer)
Amazon Bedrock Knowledgebase
Amazon ECS

6 Aurora Serverless v2

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

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

6 **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