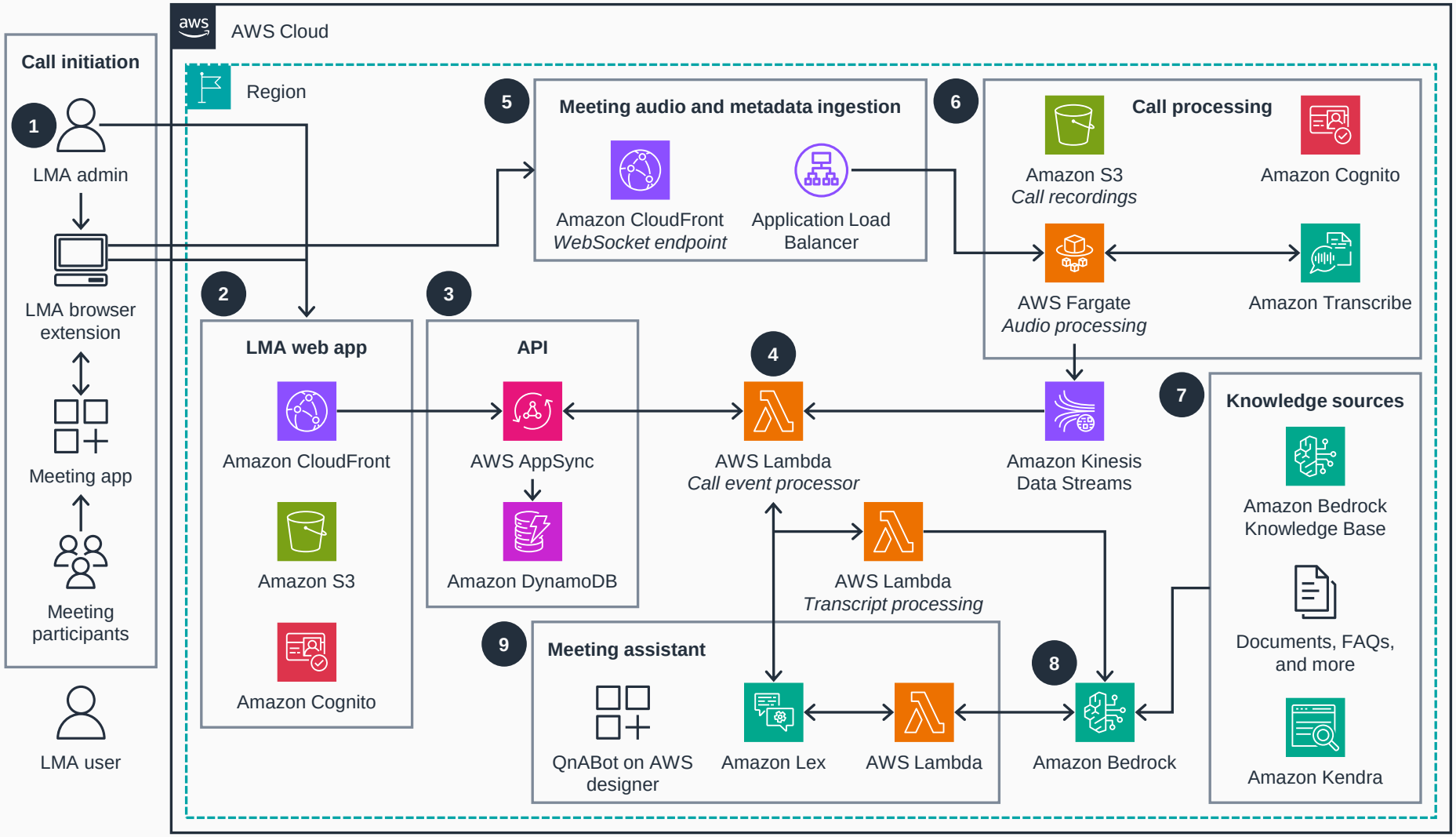
Guidance for Live Meeting Assistant on AWS

This architecture diagram shows how you can use generative AI to capture live meeting insights, summarize conversations, and translate languages.



1. Your Live Meeting Assistant (LMA) administrator installs the LMA browser extension as a plugin to their browser. Meeting participants join the meeting app.
2. The LMA bowser extension registers the meeting in the LMA web app. **Amazon CloudFront** serves the web app from an **Amazon Simple Storage Service (Amazon S3)** source. LMA users are authenticated through **Amazon Cognito**.
3. The LMA web app communicates with an API hosted by **AWS AppSync** and stores meeting information in **Amazon DynamoDB**.
4. The API is backed by **AWS Lambda**, which uses functions to process call events. Transcriptions are processed from a stream in **Amazon Kinesis Data Streams**.
5. The LMA browser extension sends audio to a **CloudFront** WebSocket endpoint. WebSocket traffic is load balanced by **Application Load Balancer**.
6. **Amazon Transcribe** transcribes the audio and stores the call recording in **Amazon S3**. Authorization occurs using **Amazon Cognito**. **AWS Fargate** clusters process the audio into a stream in **Kinesis Data Streams**.
7. The LMA solution supports retrieval-augmented generation (RAG) by enabling users to create a new (or link to an existing) Knowledge Base in **Amazon Bedrock**. Additional data sources, like those indexed in **Amazon Kendra**, can be configured in the open-source QnABot on AWS solution as a fallback RAG index.
8. **Amazon Bedrock** provides API access to LLMs to generate call summaries.
9. The LMA solution deploys open-source QnABot on AWS as a nested **AWS CloudFormation** stack for answers based on FAQs and as an orchestrator for request routing to the appropriate AI service. LMA admins interact with the chatbot using **Amazon Lex** through the LMA web app.

Reviewed for technical accuracy July 22, 2024

AWS Reference Architecture