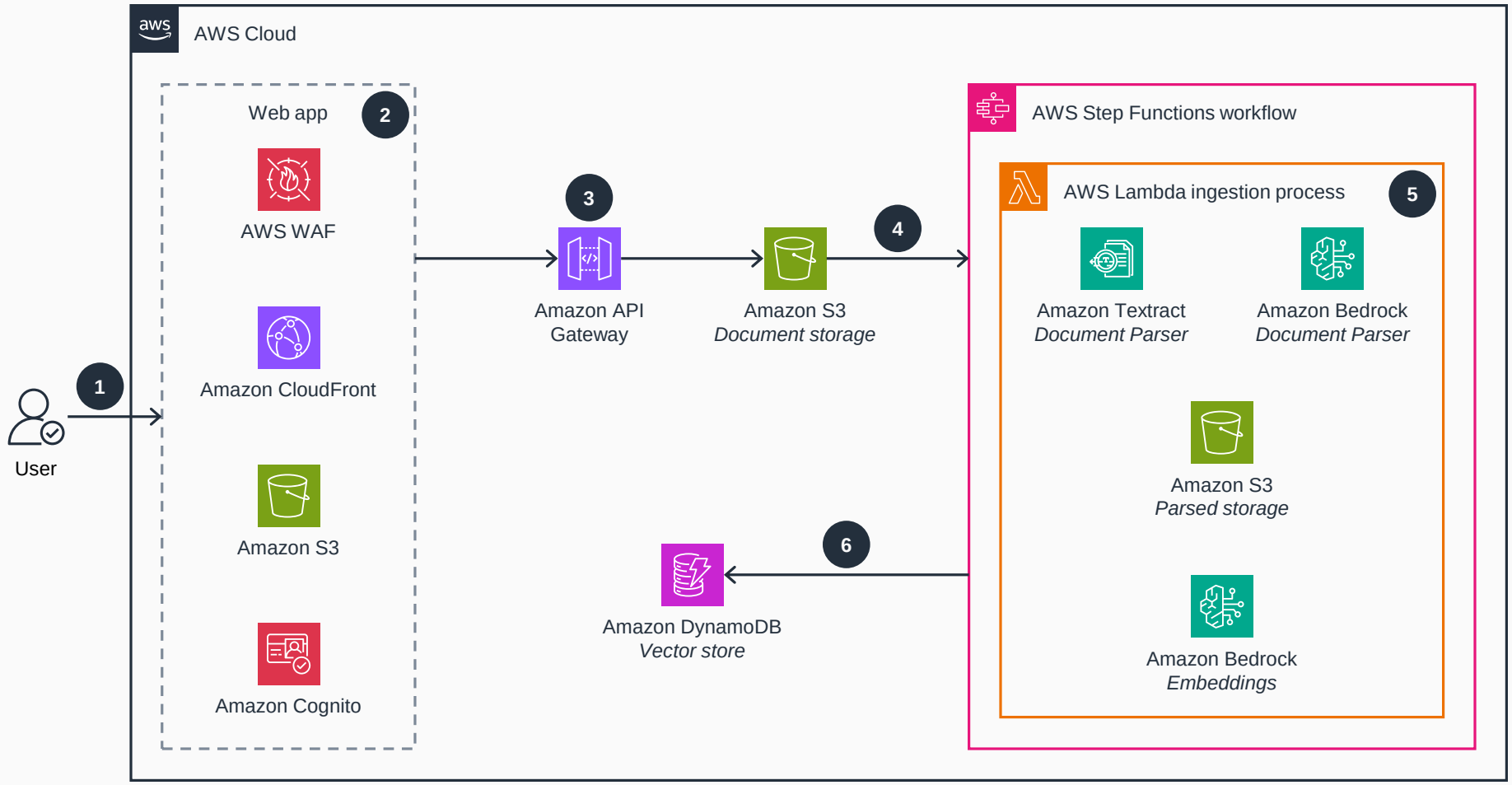
Guidance for Creating Low-Cost Semantic Search on AWS

Document ingestion and vectorization flow

This architecture diagram shows how to effectively create a low-cost vector store using Amazon DynamoDB. It shows the key components and their interactions, providing an overview of the architecture's structure and functionality. This diagram illustrates document ingestion and vectorization flow.



1. A user accesses the document upload portal through **Amazon CloudFront**, while **AWS WAF** protects against malicious traffic and common web exploits.

2. **Amazon Simple Storage Service (Amazon S3)** serves the web portal's static files (HTML, CSS, and JavaScript). **Amazon Cognito** simultaneously handles user authentication and access management.

3. **Amazon API Gateway** securely receives document uploads from the user and routes them to **Amazon S3** for encrypted storage.

4. When a new document is uploaded to **Amazon S3**, it automatically invokes an **AWS Step Functions** workflow to begin the data ingestion process.

5. Depending on the document type, a user has two options for document processing: **Amazon Textract** for text extraction and parsing or **Amazon Bedrock** for advanced document understanding. Once processed, the documents are stored in **Amazon S3**. Subsequently, an **AWS Lambda** function performs two key operations:

   a. **Text Chunking:** This operation segments the text into smaller chunks, with a user-configurable default size.

   b. **Embedding Generation:** This operation creates vector embeddings for each chunk by using **Amazon Titan** Embeddings through the **Amazon Bedrock** API.

6. The system stores document vectors in **Amazon DynamoDB** tables with separate tables for different chunk sizes to optimize retrieval performance.

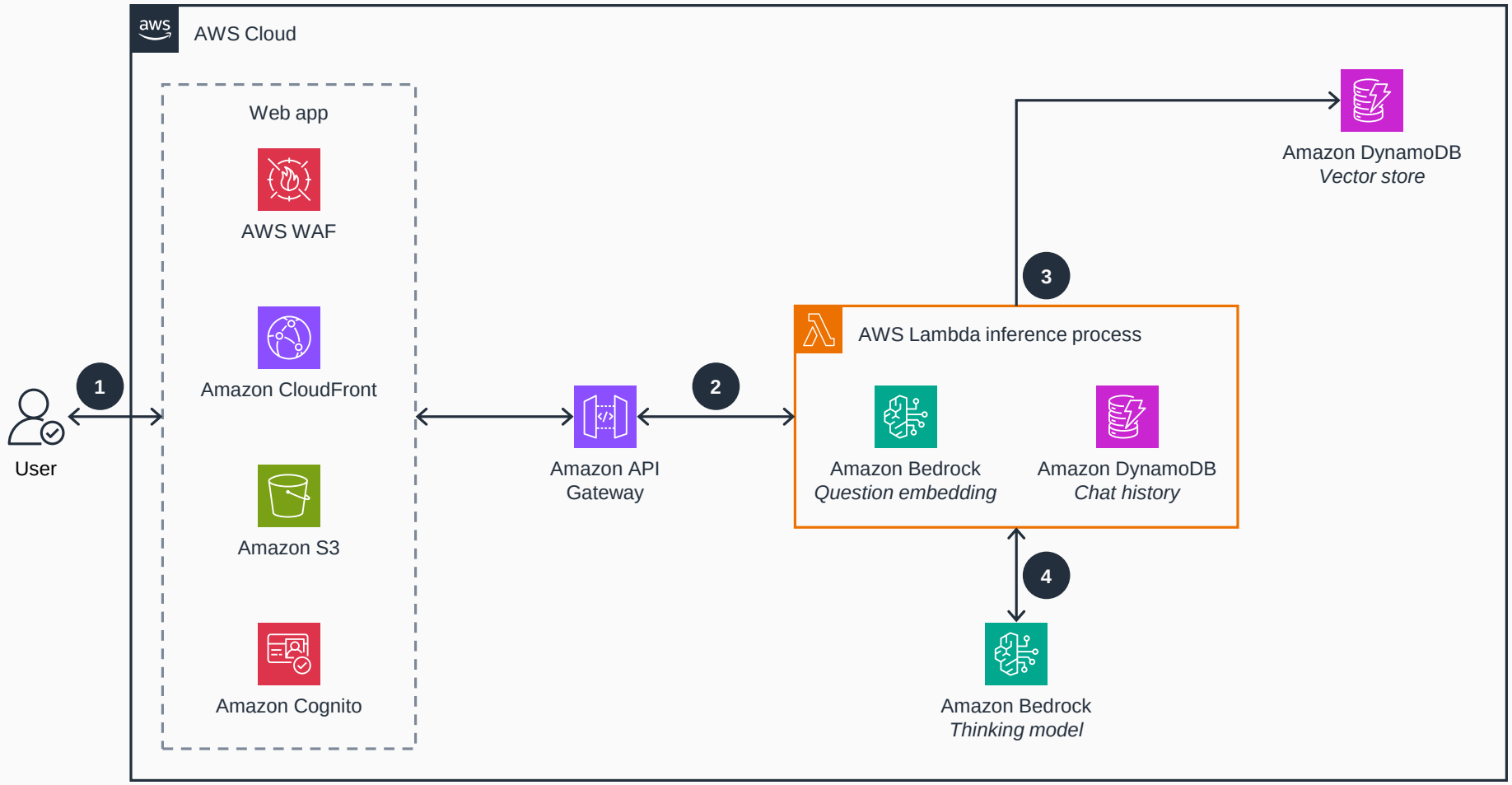
Reviewed for technical accuracy May 21, 2025

AWS Reference Architecture

Guidance for Creating Low-Cost Semantic Search on AWS

Inference flow

This architecture diagram shows how to effectively create a low-cost vector store using Amazon DynamoDB. It shows the key components and their interactions, providing an overview of the architecture's structure and functionality. This diagram illustrates inference flow.



1. A user accesses the management portal through **CloudFront**, which enables them to do the following:
   a. Adjust document processing settings (such as chunk sizes and processing models).
   b. Evaluate and test the chat interface's performance.
2. **API Gateway** routes the user's prompt to a **Lambda** function for processing.
3. The inference processes the user's prompt by converting it to vectors using an **Amazon Titan** Embeddings model. This model stores the conversation history in **DynamoDB** for context retention. Simultaneously, a **Lambda** function performs the following:
   a. It retrieves all vectors associated with the **Amazon Cognito** user from **DynamoDB**.
   b. It performs a context similarity search using inverse cosine similarity.
   c. It uses the most relevant text chunks as context for the generative AI model.
4. **Amazon Bedrock** uses Claude 3 Haiku to process the user's prompt alongside contextually relevant vectors. It then generates and delivers the response to the user through the chat interface.

Reviewed for technical accuracy May 21, 2025

AWS Reference Architecture