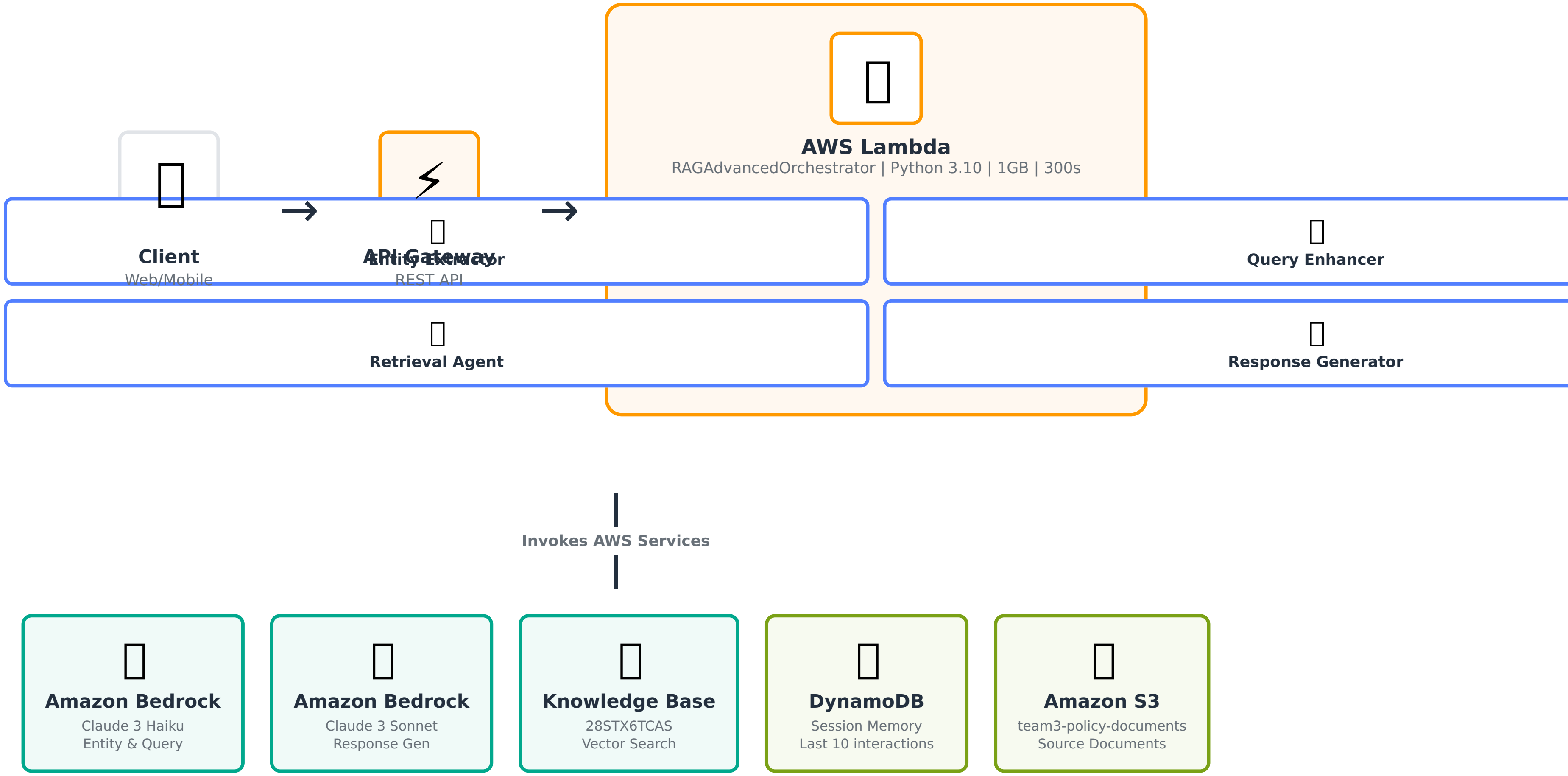
RAG Starter - Multi-Agent Architecture

Serverless RAG Application with Amazon Bedrock Knowledge Base | Region: us-east-1 | Stack: rag-api



Client
Web/Mobile

API Gateway
REST API

AWS Lambda
RAGAdvancedOrchestrator | Python 3.10 | 1GB | 300s

Entity Extractor

Query Enhancer

Retrieval Agent

Response Generator

Invokes AWS Services

Amazon Bedrock
Claude 3 Haiku
Entity & Query

Amazon Bedrock
Claude 3 Sonnet
Response Gen

Knowledge Base
28STX6TCAS
Vector Search

DynamoDB
Session Memory
Last 10 interactions

Amazon S3
team3-policy-documents
Source Documents

Request Flow

1. Client → API Gateway (POST /rag)
2. API Gateway → Lambda
3. Load session from DynamoDB
4. Extract entities (Haiku)
5. Enhance query (Haiku)
6. Retrieve docs (KB)
7. Generate response (Sonnet)
8. Save session to DynamoDB
9. Return to client

⚙ Technical Specs

Region: us-east-1
Stack: rag-api (AWS SAM)
Lambda: Python 3.10, 1024MB, 300s
Knowledge Base: 28STX6TCAS
Data Source: WZL8ZFE8LW
S3 Bucket: team3-policy-documents
DynamoDB: rag-conversation-memory
Haiku: claude-3-haiku-20240307-v1:0
Sonnet: claude-3-sonnet-20240229-v1:0

Data Source Explained

Data Source ID: WZL8ZFE8LW (S3PolicyDocs) connects the S3 bucket **team3-policy-documents** to Knowledge Base **28STX6TCAS**. When documents are uploaded to S3, this data source ingests them, creates vector embeddings, and indexes them for semantic search. The Retrieval Agent queries the Knowledge Base to fetch the top 5 most relevant documents based on the enhanced query.