Stage 1: Prerequisites & Setup

First, we set up the AWS environment.

1. **Amazon Bedrock** is enabled with models like **Anthropic Claude** (for natural language understanding) and **Amazon Titan Embeddings** (for retrieval-based tasks).
2. **IAM Roles** are configured so Bedrock can safely call other AWS services.
3. A **stock data API key** (like Alpha Vantage or Finnhub) is securely stored in **AWS Secrets Manager**.

This ensures a secure, ready environment for development.

Stage 2: Stock Data Handling & Analysis Logic

This is where the backend intelligence lives — all implemented using **AWS Lambda**.

These are representative and please look for good sources for the particular financial chatbot building.

- **Lambda #1 – Data Fetcher:**
  Fetches real-time stock data from external APIs (like Alpha Vantage).
  Input → Stock symbol (`TSLA`), Output → Latest price/time series data.
- **Lambda #2 – Technical Analyzer:**
  Performs mathematical calculations like **SMA (Simple Moving Average)**, **RSI (Relative Strength Index)**, or **EMA (Exponential Moving Average)** on that data.
- **Lambda #3 – Prediction Model:**
  Sends processed data to an **Amazon SageMaker** model to forecast future stock prices.

All raw and processed data are stored in **Amazon S3**, creating a mini **data lake** for analysis or retraining.

This stage provides the "brain" for computation and analytics.

Stage 3: Agent Orchestration (Amazon Bedrock Agent)

This is the **AI layer** — the heart of your chatbot.

- You create an **Agent in Amazon Bedrock**, powered by Anthropic Claude.
- The agent is connected to your Lambda functions through **Action Groups**

  (So the agent knows what tools it can call and what inputs to use.)

When the user asks a question:

1. The agent interprets the intent (e.g., “Fetch data”, “Analyze indicator”).
2. It automatically calls the correct Lambda function.
3. It then formulates a natural, conversational answer using the results.

This turns your logic into an intelligent conversational system.

Stage 4: User Interface (Frontend + API Gateway)

Now, users need a way to interact with your chatbot.

1. **Amazon API Gateway** is used to expose a REST API endpoint — it acts as a bridge between your web UI and the Bedrock Agent.
2. A **proxy Lambda function** connects API Gateway to the Bedrock `invoke_agent()` API.
3. On the frontend:
    - You can build a **React web app** or a **Streamlit Python app**.
    - The user types a question, which is sent to the API.
    - The response is displayed in chat format.

You can host your frontend on **Amazon S3 (static files)** and serve it securely using **Amazon CloudFront** with HTTPS.

This makes your system interactive and user-friendly.

Stage 5: Monitoring, Security & Scaling

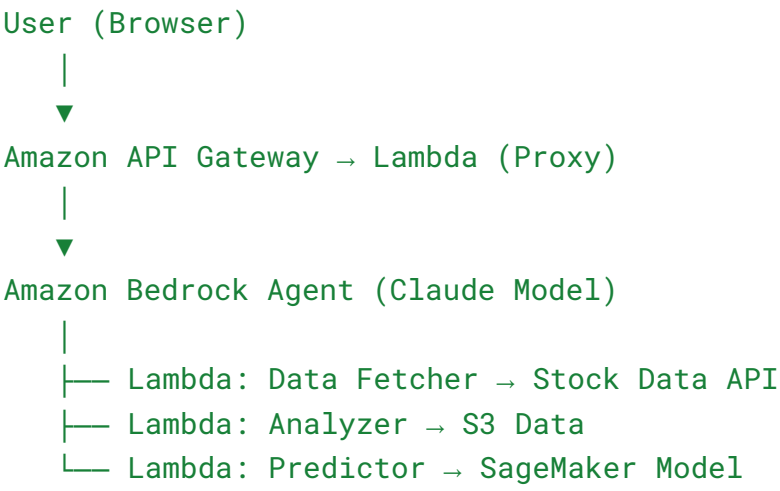
Finally, you ensure your chatbot runs smoothly, securely, and can scale.

- **CloudWatch** monitors all logs (Lambda executions, Agent responses).
- **Cognito** (optional) secures user authentication.
- **DynamoDB** stores chat history and user preferences.
- **Auto Scaling / ECS / Fargate** help the system handle more users efficiently.

This ensures the solution is reliable, secure, and production-ready.

Architecture Overview

Here's how the flow works:



Supporting Services:
S3 (data lake), CloudWatch (logs), Secrets Manager (API keys), DynamoDB (chat history).