Here’s a complete, deployable **AI-Powered FAQ Bot** for Amazon Bedrock with **Terraform IaC**, **Lambda** (Python), **API Gateway (REST)**, **S3 (FAQ data)**, plus a simple **HTML/JS** frontend.

**Refer the complete code in** bedrock-faq-bot.zip

**What you’ll deploy (at a glance)**

* **FAQ data in S3** (data/faq.json)
* **Lambda (qna\_bot)** reads FAQs from S3, fetches top-K matches with lightweight retrieval, prompts **Bedrock** (Claude/Titan/Llama) using the **Anthropic Messages API** format, and returns an answer with sources
* **API Gateway (REST)** exposes:
  + POST /ask → Lambda proxy
  + GET /health
  + OPTIONS /ask for **CORS** (preflight); Lambda also returns CORS headers.
* **Frontend**: a minimal browser UI that calls /ask.

**Step-by-step guide**

**1) Prereqs**

* Terraform ≥ 1.6, AWS CLI, Python 3.11
* An AWS account with **Amazon Bedrock** access (enable your chosen model in-region; default is **Claude 3 Sonnet**).

**2) Unzip & review**

Open the README inside the zip for a diagram and commands.

bedrock-faq-bot/

terraform/ # IaC for S3, IAM, Lambda, API GW (REST + CORS)

lambda/qna.py # Lambda handler (retrieval + Bedrock prompt)

web/index.html # Simple chat UI (run locally)

data/faq.json # Sample FAQ file

tools/upload\_faq.py# Helper to upload FAQ JSON to S3

**3) Deploy infrastructure**

cd terraform

terraform init

terraform apply -auto-approve

Terraform creates:

* **S3** bucket (private, SSE-S3) for FAQs
* **IAM** role with least-privilege including bedrock:InvokeModel
* **Lambda** (Python 3.11)
* **API Gateway (REST)** resources + methods with **Lambda proxy** integration and a **MOCK OPTIONS** method to enable CORS for /ask.

Outputs include:

* faq\_bucket\_name
* api\_base\_url (e.g., https://xxxx.execute-api.<region>.amazonaws.com/prod)

**4) Upload your FAQ data**

Edit data/faq.json or keep the sample, then upload:

# Using the helper script

python ../tools/upload\_faq.py --bucket <faq\_bucket\_name>

# Or via AWS CLI

aws s3 cp ../data/faq.json s3://<faq\_bucket\_name>/data/faq.json

**5) Test the API**

API="https://xxxx.execute-api.<region>.amazonaws.com/prod"

curl -s -X POST "$API/ask" \

-H "content-type: application/json" \

-d '{"question":"What is the return policy?"}'

**6) Try the frontend**

Open web/index.html in your browser, paste the **API Base** (.../prod) in the header, then ask a question.

**How it works (under the hood)**

1. **Retrieval (lightweight)**
   * Lambda loads faq.json from S3 on cold start and caches it in memory.
   * For each query, it computes a simple similarity (Jaccard + substring boost) against FAQ Q/A text and chooses top-K (default 5).
2. **Prompting Bedrock**
   * Lambda crafts an **Anthropic Messages API** request with:
     + anthropic\_version: "bedrock-2023-05-31"
     + messages: [{role: "user", content: [{ type:"text", text: "...context + user question..." }]}]
   * Calls InvokeModel on your chosen model (default Claude 3 Sonnet) and returns the text.
3. **API Gateway (REST) + CORS**
   * POST /ask uses **Lambda proxy** integration; Lambda adds Access-Control-Allow-\* headers.
   * OPTIONS /ask (MOCK) returns the CORS headers for preflight as recommended.

**Files you’ll likely tweak**

* terraform/variables.tf
  + region (defaults to ap-south-1)
  + bedrock\_model\_id (try a newer **Claude 3.5** Sonnet/Haiku if enabled in your account/region).
* data/faq.json
  + Your real FAQs (Q/A pairs). Keep answers concise; the bot quotes from these.
* lambda/qna.py
  + Retrieval strategy or prompt wording
  + MAX\_TOKENS / TEMPERATURE

**Security & production notes**

* **Privacy**: S3 bucket is private with **SSE-S3**. Consider **SSE-KMS**, VPC endpoints, and CloudFront for the UI if you host it.
* **Scale**: For larger KBs, add real retrieval: **Titan Embeddings** + OpenSearch/Kendra (not included here to keep the stack minimal).
* **CORS**: Implemented both at API Gateway (OPTIONS MOCK) and Lambda headers to accommodate browser preflight.
* **Costs**: Bedrock tokens, Lambda, API Gateway, S3. Destroy when done.

**Architecture diagram**

User Browser ──POST /ask──▶ API Gateway (REST) ──Lambda Proxy──▶ Lambda (qna\_bot)

▲ │

│ │ InvokeModel

└────── HTML/JS fetch ◀─────────────── CORS ───────────────┘ to Bedrock

│

└────────── Reads FAQ JSON from S3