**Text Summarization Tool**

* **Goal:** Summarize PDFs or text content.
* **Tech:** Bedrock + AWS Textract (for OCR).
* **Steps:**
  1. Extract text from PDF using Textract.
  2. Pass to Bedrock for summarization.
  3. Output to S3 or return via API.
* **Use Case:** Meeting note summaries, legal document briefs.

**Download the complete code -** bedrock-text-summarizer.zip

**What you’ll deploy**

* **Upload PDFs → S3** → **Lambda (ingest)** → **Textract (async)** → **SNS → SQS** → **Lambda (postprocess: fetch pages, summarize with Bedrock, write to S3)**
* **Raw text API**: **API Gateway (HTTP)** → **Lambda (text\_summarizer)** → **Bedrock** → JSON response
* PDFs require Textract’s **asynchronous** APIs: StartDocumentTextDetection → GetDocumentTextDetection.
* Job completion is delivered via **NotificationChannel** to an **SNS** topic (Textract assumes a role to publish).
* Bedrock calls use **InvokeModel** with the **Anthropic Messages API** payload (anthropic\_version: "bedrock-2023-05-31").
* The default model in the IaC is **Claude 3 Sonnet** (anthropic.claude-3-sonnet-20240229-v1:0), widely available (including **ap-south-1**). You can switch to Claude 3.5 if enabled in your account/region.

**Quick Start (step-by-step)**

1. **Prereqs**
   * Terraform ≥ 1.6, AWS CLI, Python 3.11
   * An AWS account with access to **Amazon Bedrock** in your chosen **region** (defaults to **ap-south-1**). Enable the model in the Bedrock console if you haven’t already.
2. **Unzip & review**
   * Open the README inside the zip for an architecture diagram and commands.
3. **Deploy**
   * cd terraform
   * terraform init
   * terraform apply -auto-approve

Terraform stands up:

* + **S3** input/output buckets (SSE-S3)
  + **SNS** topic + **SQS** queue (raw delivery on)
  + **IAM**: Lambda roles, **Textract publish role** that Textract assumes to post to SNS
  + **Lambda**: pdf\_ingest, textract\_postprocess, text\_summarizer
  + **S3 → Lambda** trigger (new PDFs) and **SQS → Lambda** event source mapping
  + **API Gateway (HTTP)**: POST /summarize → text\_summarizer

IaC uses official Terraform resources for **S3 bucket notifications** and **Lambda event source mapping**.

The SNS→SQS subscription sets RawMessageDelivery=true so your postprocessor reads the **Textract JSON directly** from SQS message bodies.

1. **Test (PDF path)**
   * Terraform output shows input\_bucket\_name and output\_bucket\_name.
   * Upload a PDF under the incoming/ prefix:
   * aws s3 cp sample.pdf s3://<input-bucket>/incoming/sample.pdf
   * Within ~seconds–minutes (depends on pages), find outputs:
     + s3://<output-bucket>/extracted/sample.txt (all OCR’d text)
     + s3://<output-bucket>/summaries/sample.summary.txt (AI summary)
2. **Test (raw text API)**
   * Terraform output shows api\_invoke\_url:
   * curl -s -X POST "$API/prod/summarize" \
   * -H "content-type: application/json" \
   * -d '{"text":"Minutes: Team discussed Q4 targets..."}' | jq
3. **Tear down**
   * terraform destroy

**What’s inside the zip**

* **terraform/**
  + main.tf, variables.tf (region defaults to ap-south-1), and outputs
  + Creates S3, SNS, SQS, IAM, Lambda, S3 notification, SQS mapping, and HTTP API
* **lambda/**
  + pdf\_ingest.py — S3 trigger; calls StartDocumentTextDetection with NotificationChannel (SNS + role)
  + textract\_postprocess.py — SQS trigger; paginates GetDocumentTextDetection, summarizes via Bedrock, writes summary & raw text to S3
  + text\_summarizer.py — HTTP API Lambda; summarizes raw text with Bedrock
* **README.md** — architecture diagram, commands, notes

The Bedrock calls are implemented with **boto3** bedrock-runtime.invoke\_model using the **Anthropic Messages API** format; you can swap model IDs via Terraform variable bedrock\_model\_id.

**Production notes & options**

* **Model choice**: Keep anthropic.claude-3-sonnet-20240229-v1:0 default for broad regional support; upgrade to **Claude 3.5 Sonnet/Haiku** if available and enabled in your region.
* **Large docs**: Textract is already async for PDFs; the postprocessor paginates through results until NextToken is exhausted.
* **Hardening**: Add VPC endpoints, **KMS (SSE-KMS)** for S3, DLQ for the SQS→Lambda mapping, and IAM boundary policies.
* **Alternative flow**: Replace SNS+SQS with **Step Functions** if you prefer orchestration; Textract async remains the same.
* **Costs**: Textract (per page), Bedrock (per token), Lambda/SNS/SQS/S3; destroy when done.