AWS, If I have a number of .txt and.pdf text documents how do I produce aBedrock Knowledge base?

**ChatGPT said:**

To build an **Amazon Bedrock Knowledge Base** from a collection of .txt and .pdf documents, you'll follow a process that includes data preparation, ingestion, chunking and embedding, and finally integrating it with a Bedrock agent for retrieval-augmented generation (RAG). Here's a **detailed step-by-step guide**:

**🧱 Overview: What is a Bedrock Knowledge Base?**

A **Bedrock Knowledge Base** allows you to augment LLM outputs using your own **unstructured or semi-structured documents**. It works by:

1. Splitting your content into chunks.
2. Embedding those chunks into a **vector store** using a **Foundation Model (FM)**.
3. At query time, retrieving relevant chunks and passing them to the FM to generate responses.

**📂 Step 1: Prepare Your Documents**

You have .txt and .pdf files. To use them:

1. **Upload to Amazon S3**:
   * Organize your documents into a single S3 bucket or a folder within a bucket.
2. **Convert PDFs to Text (Optional but Recommended)**:
   * You can extract text from PDFs using tools like:
     + Amazon Textract (managed service, better for scanned PDFs)
     + PyMuPDF, pdfminer.six, or pdfplumber in Python for simple PDFs.

💡 If the PDFs are text-based (not scanned images), you can skip Textract.

**🧠 Step 2: Create a Bedrock Knowledge Base**

1. **Go to AWS Console → Amazon Bedrock → Knowledge Bases**
2. Click **"Create Knowledge Base"**.
3. Define the following:
   * **Name & Description**
   * **S3 Data Source**:
     + Point to the S3 path where your .txt and .pdf files are stored.
   * **Data Ingestion Type**: Choose **batch ingestion** (initial population).
   * **Embedding Model**: Choose one (e.g., Titan Embeddings G1 or Cohere Embed).
   * **Vector Store**: Choose between built-in (Amazon OpenSearch Serverless) or custom (like Pinecone or Redis Enterprise).

**⚙️ Step 3: Ingest and Process Data**

* Once the Knowledge Base is created, click **"Ingest documents"**.
* Bedrock will:
  + Split large documents into **manageable text chunks**.
  + Call the embedding model on each chunk.
  + Store the resulting vectors in the vector store.

You can monitor the ingestion process from the console.

**🤖 Step 4: Connect to a Bedrock Agent**

If you want to use your Knowledge Base for question answering:

1. Go to **Amazon Bedrock → Agents**
2. Create or update an Agent.
3. Attach your Knowledge Base in the **"Knowledge Base"** section.
4. The agent will now:
   * Receive a user question.
   * Query your Knowledge Base to find relevant chunks.
   * Feed those chunks into the FM as context.
   * Return a grounded answer.

**🛠️ Advanced: Automation via SDK**

You can automate all of this using **AWS SDK (Python boto3)**, with features like:

* Batch document upload.
* Periodic ingestion with updated documents.
* Triggering ingestion via Lambda or S3 events.

**📌 Best Practices**

* **Chunk size**: Bedrock handles this, but ensure large PDFs are logically segmented.
* **Document updates**: Re-ingest when updating or adding files.
* **Metadata tagging**: Add custom metadata in filenames or S3 object tags for filtering later.

**✅ Summary**

| **Step** | **Action** |
| --- | --- |
| 1 | Upload .txt/.pdf files to S3 |
| 2 | Create Bedrock Knowledge Base (select embedding model & vector store) |
| 3 | Ingest documents (Bedrock handles chunking & embedding) |
| 4 | Attach Knowledge Base to an Agent for RAG |
| 5 | Query via Agent or API |

Would you like a sample script to convert PDFs and upload them to S3, or a walkthrough using the AWS SDK (boto3) to automate the process?

Top of Form

Bottom of Form

I like like a sample script to convert PDFs and upload them to S3, or a walkthrough using the AWS SDK (boto3) to automate the process

**ChatGPT said:**

Great! Here's a **complete workflow** with:

1. ✅ A **Python script** that:
   * Extracts text from .pdf files (using pdfplumber).
   * Saves them as .txt.
   * Uploads both .pdf and .txt files to S3.
2. ✅ A **boto3-based automation** to trigger ingestion into a Bedrock Knowledge Base (assuming it's already set up).

**🧰 Prerequisites**

Install required libraries:

bash

CopyEdit

pip install boto3 pdfplumber

**📄 Step 1: Python Script – Convert and Upload PDFs to S3**

python

CopyEdit

import os

import boto3

import pdfplumber

# Configuration

AWS\_REGION = 'us-east-1'

S3\_BUCKET = 'your-bucket-name'

S3\_PREFIX = 'bedrock-docs/' # Optional folder path inside the bucket

LOCAL\_FOLDER = 'pdf\_docs' # Folder with your PDF files

s3 = boto3.client('s3', region\_name=AWS\_REGION)

def convert\_pdf\_to\_txt(pdf\_path):

txt\_path = pdf\_path.replace('.pdf', '.txt')

with pdfplumber.open(pdf\_path) as pdf:

text = '\n'.join(page.extract\_text() or '' for page in pdf.pages)

with open(txt\_path, 'w', encoding='utf-8') as f:

f.write(text)

return txt\_path

def upload\_file\_to\_s3(file\_path, s3\_key):

s3.upload\_file(file\_path, S3\_BUCKET, s3\_key)

print(f"Uploaded: {s3\_key}")

def process\_pdfs():

for filename in os.listdir(LOCAL\_FOLDER):

if filename.lower().endswith('.pdf'):

full\_pdf\_path = os.path.join(LOCAL\_FOLDER, filename)

print(f"Processing: {filename}")

# Convert to TXT

txt\_path = convert\_pdf\_to\_txt(full\_pdf\_path)

# Upload both PDF and TXT

upload\_file\_to\_s3(full\_pdf\_path, S3\_PREFIX + filename)

upload\_file\_to\_s3(txt\_path, S3\_PREFIX + os.path.basename(txt\_path))

process\_pdfs()

☑️ You can now point your Bedrock Knowledge Base to the S3 path s3://your-bucket-name/bedrock-docs/.

**🤖 Step 2: Automate Ingestion with boto3 (Optional)**

Once you've uploaded or updated documents, you can trigger ingestion like this:

python

CopyEdit

import boto3

bedrock = boto3.client('bedrock-agent', region\_name='us-east-1')

# Replace with your actual values

knowledge\_base\_id = 'kb-xxxxxxxxxx' # Get from AWS Console

data\_source\_id = 'ds-xxxxxxxxxx' # Get from AWS Console

def start\_ingestion():

response = bedrock.start\_ingestion\_job(

knowledgeBaseId=knowledge\_base\_id,

dataSourceId=data\_source\_id

)

print(f"Ingestion started: {response['ingestionJob']['ingestionJobId']}")

start\_ingestion()

📝 You can find your Knowledge Base ID and Data Source ID in the Bedrock console or list them via list\_knowledge\_bases() and list\_data\_sources().

**🔁 Optional: Run Regular Syncs via Lambda or Cron**

* **Lambda**: Use the PDF conversion + S3 upload script in a Lambda function that is triggered on S3 upload.
* **Cron job**: Schedule it locally or on an EC2 instance with cron to run every night or week.

**✅ Final Notes**

* You don’t have to store both .pdf and .txt in S3 – you can skip uploading PDFs if only .txt is needed.
* Bedrock will read .txt files directly without needing a preprocessor.

Would you like a version of this that runs entirely in a **Jupyter notebook** (e.g., in SageMaker)?

Top of Form

Bottom of Form

I like a version of this that runs entirely in a Jupyter notebook (e.g., in SageMaker)

**ChatGPT said:**

Perfect! Below is a full **Jupyter Notebook-compatible workflow** that you can run inside **Amazon SageMaker Studio or JupyterLab**. It:

1. Extracts text from PDF files using pdfplumber.
2. Saves .txt files locally.
3. Uploads both .pdf and .txt to S3.
4. Optionally triggers ingestion in an existing **Amazon Bedrock Knowledge Base**.

**📓 Jupyter Notebook: PDF → S3 → Bedrock Knowledge Base**

**🔧 Step 1: Install Dependencies (if needed)**

python

CopyEdit

!pip install -q boto3 pdfplumber

**📁 Step 2: Set Up Configuration**

python

CopyEdit

import os

import boto3

# --- CONFIG ---

AWS\_REGION = 'us-east-1' # Or your region

S3\_BUCKET = 'your-s3-bucket-name'

S3\_PREFIX = 'bedrock-docs/' # Optional folder inside the bucket

LOCAL\_FOLDER = 'pdf\_docs' # Local folder with .pdf files

UPLOAD\_TXT = True # Whether to convert/upload .txt too

# Optional: for triggering ingestion

TRIGGER\_INGESTION = True

KNOWLEDGE\_BASE\_ID = 'kb-xxxxxxxxx' # Replace with your Knowledge Base ID

DATA\_SOURCE\_ID = 'ds-xxxxxxxxx' # Replace with your Data Source ID

# Initialize clients

s3\_client = boto3.client('s3', region\_name=AWS\_REGION)

bedrock = boto3.client('bedrock-agent', region\_name=AWS\_REGION)

**🧪 Step 3: PDF → TXT Conversion and Upload to S3**

python

CopyEdit

import pdfplumber

def convert\_pdf\_to\_txt(pdf\_path):

txt\_path = pdf\_path.replace('.pdf', '.txt')

with pdfplumber.open(pdf\_path) as pdf:

text = '\n'.join(page.extract\_text() or '' for page in pdf.pages)

with open(txt\_path, 'w', encoding='utf-8') as f:

f.write(text)

return txt\_path

def upload\_file\_to\_s3(file\_path, s3\_key):

s3\_client.upload\_file(file\_path, S3\_BUCKET, s3\_key)

print(f"Uploaded: s3://{S3\_BUCKET}/{s3\_key}")

def process\_pdfs():

for filename in os.listdir(LOCAL\_FOLDER):

if filename.lower().endswith('.pdf'):

full\_pdf\_path = os.path.join(LOCAL\_FOLDER, filename)

print(f"Processing: {filename}")

# Convert to text

if UPLOAD\_TXT:

txt\_path = convert\_pdf\_to\_txt(full\_pdf\_path)

upload\_file\_to\_s3(txt\_path, S3\_PREFIX + os.path.basename(txt\_path))

# Upload original PDF

upload\_file\_to\_s3(full\_pdf\_path, S3\_PREFIX + filename)

process\_pdfs()

**⚙️ Step 4: Trigger Bedrock Ingestion (Optional)**

python

CopyEdit

def start\_ingestion():

response = bedrock.start\_ingestion\_job(

knowledgeBaseId=KNOWLEDGE\_BASE\_ID,

dataSourceId=DATA\_SOURCE\_ID

)

job\_id = response['ingestionJob']['ingestionJobId']

print(f"Ingestion started (Job ID: {job\_id})")

return job\_id

if TRIGGER\_INGESTION:

start\_ingestion()

**✅ Summary**

You now have a notebook that:

* Works end-to-end in SageMaker Studio.
* Converts PDFs → .txt → uploads both to S3.
* Optionally triggers ingestion into a Bedrock knowledge base.

**🚀 Optional Enhancements**

* **Status Polling**:

python

CopyEdit

def check\_ingestion\_status(job\_id):

status = bedrock.get\_ingestion\_job(

knowledgeBaseId=KNOWLEDGE\_BASE\_ID,

ingestionJobId=job\_id

)

print(status['ingestionJob']['status'])

check\_ingestion\_status('your-job-id-here')

* **Auto-detect & skip already-uploaded files**.
* **Preprocess chunking manually** for better control before Bedrock ingestion.