

Project: Intelligent Document Querying System

1. Project Overview

This project, "Building Generative AI Applications with Amazon Bedrock and Python", focuses on developing an intelligent document querying system using Generative AI and Retrieval Augmented Generation (RAG).

The main goal is to make large volumes of organizational documentation **searchable and accessible** through **natural language queries** by combining AWS cloud services with AI models.

Key services used:

- **Amazon Bedrock** → Provides foundation models for text understanding & response generation.

- **Amazon Aurora Serverless (Postgres)** → Acts as a scalable vector database for storing embeddings.
- **Amazon S3** → Stores PDF documents that serve as the knowledge base.
- **Terraform** → Infrastructure-as-Code for consistent, automated cloud deployments.
- **Python scripts** → Handle document ingestion, uploading, and querying.

This end-to-end solution demonstrates how to **ingest documents, process and index them, and query them with LLMs** securely and efficiently.

Base infrastructure creation

Deployment of VPC, Aurora Postgres Serverless, and S3 bucket using Terraform

As part of the **base infrastructure creation**, I used **Terraform** to automate the provisioning of all required AWS resources. The deployment included:

- **VPC** → A Virtual Private Cloud was created with proper CIDR block allocation to securely host resources.
- **Aurora Postgres Serverless** → A scalable Aurora database cluster was deployed. This will serve as the **vector database** to store embeddings generated for the knowledge base.
- **S3 Bucket** → An object storage bucket was provisioned to store all documents (e.g., PDFs) that form the knowledge corpus for the Bedrock Knowledge Base.

After running:

```
terraform init terraform
```

```
apply
```

Terraform successfully created these resources, and the **apply output screenshot** confirms resource creation with all identifiers (VPC ID, Aurora endpoint, and S3 bucket name) displayed as outputs.

Result:

The infrastructure was successfully deployed, verified, and is ready for use in later stages of the project (knowledge base integration and querying).

The screenshot displays a developer's environment with several open windows:

- File Explorer:** Shows the project structure for "cd13926-Building-Generative-AI-Applications-with-Amazon-Bedrock-and-Python-project-solution". The `stack1` folder is expanded, showing files like `.terraform`, `.terraform.lock.hcl`, `main.tf`, `terraform.tstate`, `variables.tf`, and `venv`.
- Terminal:** Multiple PowerShell and Python terminals are running, with one terminal showing Terraform code for setting up an Aurora Serverless cluster.
- CloudShell:** An AWS CloudShell window titled "bedrock-kb-133720367604" is open, showing the AWS Lambda function code.
- Amazon S3:** A browser-based interface for the "documents" bucket. It lists objects such as `bulldozer-bd850-spec-sheet.pdf`, `dump-truck-dt1000-spec-sheet.pdf`, `excavator-eg50-spec-sheet.pdf`, `forklift-fl250-spec-sheet.pdf`, and `mobile-crane-mc750-spec-sheet.pdf`, all in PDF format.

PS D:\udacity\cd13926-Building-Generative-AI-Applications-with-Amazon-Bedrock-and-Python-project-solution\stack2> \$env:PATH += ";\$env:USERPROFILE\terraform"; terraform output

s3_bucket_name = "bedrock-kb-133720367604"

stack2_status = "Stack 2 completed - Documents uploaded to S3"

uploaded_documents = [

"documents/bulldozer-bd850-spec-sheet.pdf",
"documents/dump-truck-dt1000-spec-sheet.pdf",
"documents/excavator-x950-spec-sheet.pdf",
"documents/forklift-fl250-spec-sheet.pdf",
"documents/mobile-crane-mc750-spec-sheet.pdf",
]

PS D:\udacity\cd13926-Building-Generative-AI-Applications-with-Amazon-Bedrock-and-Python-project-solution\stack2> \stack1> terraform apply -auto-approve

aurora_arn = "arn:aws:rds:us-west-2:133720367604:cluster:my-aurora-serverless"

aurora_endpoint = "my-aurora-serverless.cluster-cu2bffdza994.us-west-2.rds.amazonaws.com"

db_endpoint = "my-aurora-serverless.cluster-cu2bffdza994.us-west-2.rds.amazonaws.com"

db_reader_endpoint = "my-aurora-serverless.cluster-ro-cu2bffdza994.us-west-2.rds.amazonaws.com"

private_subnet_ids = [

"subnet-010f91677bcca70d7",
"subnet-0a1b9b2ee321af52b",
"subnet-08f8ebdea37df3806",
]

rds_secret_arn = "arn:aws:secretsmanager:us-west-2:133720367604:secret:my-aurora-serverless-3yBTSS"

s3_bucket_name = "arn:aws:s3:::bedrock-kb-133720367604"

vpc_id = "vpc-0d24aad5de0cac8e5"

PS D:\udacity\cd13926-Building-Generative-AI-Applications-with-Amazon-Bedrock-and-Python-project-solution\stack2> terraform apply -auto-approve

Outputs:

s3_bucket_name = "bedrock-kb-133720367604"

stack2_status = "Stack 2 completed - Documents uploaded to S3"

uploaded_documents = [

"documents/bulldozer-bd850-spec-sheet.pdf",
"documents/dump-truck-dt1000-spec-sheet.pdf",
"documents/excavator-x950-spec-sheet.pdf",
"documents/forklift-fl250-spec-sheet.pdf",
"documents/mobile-crane-mc750-spec-sheet.pdf",
]

PS D:\udacity\cd13926-Building-Generative-AI-Applications-with-Amazon-Bedrock-and-Python-project-solution\stack2> aws s3 ls s3://bedrock-kb-133720367604/documents/

Date	Size	Name
2025-10-01 15:42:32	459443	bulldozer-bd850-spec-sheet.pdf
2025-10-01 15:42:32	366473	dump-truck-dt1000-spec-sheet.pdf
2025-10-01 15:42:32	280903	excavator-x950-spec-sheet.pdf
2025-10-01 15:42:32	232388	forklift-fl250-spec-sheet.pdf
2025-10-01 15:42:32	366291	mobile-crane-mc750-spec-sheet.pdf

Bedrock Knowledge Base Demo Setup

=====
 Bedrock Claude 3 model access confirmed!
 Response: Heavy machinery generally refers to large, powerful equipment and vehicles used for industrial, cons...
 Demo Knowledge Base configuration created:
 KB ID: DEMO-KB-HEAVY-MACHINERY-2025
 Documents: 5 PDFs

Demo setup complete!

Focus folder in explorer (ctrl + click) [ERY-2025]
 PS D:\udacity\cd13926-Building-Generative-AI-Applications-with-Amazon-Bedrock-and-Python-project-solution> aws sts get-caller-identity
 {
 "UserId": "AROAR6IS24H2B5X6TE3BN:user4321795=8afbf224-be41-4c94-a358-2ae03fba2edf",
 "Account": "133720367604",
 "Arn": "arn:aws:sts::133720367604:assumed-role/voclabs/user4321795=8afbf224-be41-4c94-a358-2ae03fba2edf"
}

Amazon Bedrock > Knowledge Bases

Introducing S3 Vectors as a vector store (currently in preview)
 S3 vector buckets are optimized for durable and cost-effective storage of large long-term data sets while maintaining subsecond query performance. This feature is currently in preview and we don't recommend using it for production workloads.

Preparing vector database in Amazon Aurora PostgreSQL Serverless ("KnowledgeBaseQuickCreateAurora-a33740eb-b640-4f29-a859-7f10d54908f0"). This process may take up to 13 minutes to complete.

Knowledge Bases

Chat with your document

How it works

1. Create a Knowledge Base with

2. Test the Knowledge Base

3. Use the Knowledge Base

Knowledge Bases

Find Knowledge Base

Name	Status	Type	Data sources	Description	Creation time	Last sync date	Last sync warnings
Heavy_Machinery_Specifications	Available	S3	s3://bedrock-k...		October 01, 2025, 16:30 (UTC+05:30)		

Heavy_Machinery_Specifications

Test Knowledge Base Delete Edit

Knowledge Base overview

Knowledge Base name: Heavy_Machinery_Specifications
 Knowledge Base ID: L3CRT5Q79H
 Knowledge Base description: Heavy Machinery Specifications
 Service Role: AmazonBedrockExecutionRoleForKnowledgeBase_jhhvf
 Created date: October 01, 2025, 16:30 (UTC+05:30)

Log Deliveries
 Configure log deliveries and event logs in the Edit page.

Retrieval-Augmented Generation (RAG) type
 Vector store

Data source (1)
 Data sources contain information returned when querying a Knowledge Base.

Find data source	Data source...	Status	Data source ty...	Account ID	Source Link	Last sync time	Last sync war...	Chunking stra...	Parsing strategy	Data deletion ...
knowledge-b...	Available	S3	13372036760...	s3://bedrock-k...	-	-	-	Default	Default	Delete

Tags (3)
 A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value
aws:cloudformation:stack-id	arn:aws:cloudformation:us-west-2:133720367604:stack/KnowledgeBaseQuickCreateAurora-a33740eb-b640-4f29-a859-7f10d54908f0/dd186230-9eb4-11f0-9505-02963532d6d7
aws:cloudformation:stack-name	KnowledgeBaseQuickCreateAurora-a33740eb-b640-4f29-a859-7f10d54908f0

Screenshot of the AWS Secrets Manager console showing a list of secrets in the 'us-west-2' region. The table includes columns for Secret name, Description, Last retrieved (UTC), and Actions.

Secret name	Description	Last retrieved (UTC)
rdscluster-771860db-008e-4ece-8e62-787c48280a8b	The secret associated with the primary RDS DB cluster: arn:aws:rds:us-west-2:133720367604:cluster:knowledgebasequickcreateaurora-a33-auroraldbcluster-7a10c285ff	October 1, 2025
BedrockUserSecret-Sub4qPobtOV	Secret required for Bedrock Knowledge Base to interact with Aurora Cluster	October 1, 2025
my-aurora-serverless		October 1, 2025

Screenshot of the AWS Secrets Manager console showing the details for the secret 'rdscluster-771860db-008e-4ece-8e62-787c48280a8b'. The 'Secret details' tab is selected, showing the secret's description, ARN, and creation date.

Secret details

Secret description: The secret associated with the primary RDS DB cluster: arn:aws:rds:us-west-2:133720367604:cluster:knowledgebasequickcreateaurora-a33-auroraldbcluster-7a10c285ff

Secret ARN: arn:aws:secretsmanager:us-west-2:133720367604:secret:rdscluster-771860db-008e-4ece-8e62-787c48280a8b

Created: October 1, 2025

Secret value

Resource permissions - optional

Screenshot of the AWS VPC console showing the details for the VPC 'vpc-0d24aad5de0cac8e5'. The 'Details' tab is selected, displaying information such as VPC ID, State, and Network ACL settings.

Details

VPC ID: vpc-0d24aad5de0cac8e5

State: Available

Main network ACL: ad-5eef0f07534bb82c / bedrock-poc-vpc-default

IPv4 CIDR: 10.0.0.0/16

Block Public Access: Off

DHCP options set: dhcp-options-1998070208 / bedrock-poc-vpc-dhcp

Default VPC: No

Network Address Usage metrics: Disabled

IPv6 CIDR: 10.0.0.0/16

Route 53 Resolver DNS Firewall rule groups: Failed to list rule groups

Block Public Access: Off

DHCP options set: dhcp-options-1998070208 / bedrock-poc-vpc-dhcp

Main route table: rtb-0fca5ae998070208 / bedrock-poc-vpc-default

IPv6 pool: -

Owner ID: 133720367604

Resource map

Your VPC: bedrock-poc-vpc

Subnets (3): us-west-2a, us-west-2b, us-west-2c

Route tables (4): bedrock-poc-vpc-private-us-west-2a, bedrock-poc-vpc-private-us-west-2b, bedrock-poc-vpc-us-west-2a, bedrock-poc-vpc-us-west-2b

Network Connections (0):

Proper configuration and security settings

Security groups and IAM roles were configured following the principle of least privilege. The Aurora database only allows inbound traffic on port 5432. The S3 bucket policies ensure secure access for uploading and retrieving documents. Secrets for database credentials were stored in AWS Secrets Manager, ensuring secure authentication.

Secrets		Store a new secret
Secret name	Description	Last retrieved (UTC)
rdscluster-a475ba15-7d2d-4d49-a440-faf29eaad18a	The secret associated with the primary RDS DB cluster: arn:aws:rds:us-west-2:133720367604:cluster:knowledgebasequickcreateaurora-473:auroradbcluster-2litzgruuv0y	September 29, 2025
BedrockUserSecret-pRiTetW3arZ3	Secret required for Bedrock Knowledge Base to interact with Aurora Cluster	September 29, 2025
my-aurora-serverless	-	September 29, 2025

Database properly configured for vector storage.

Using the script `scripts/aurora_sql.sql`, I enabled Postgres extensions required for vector similarity search (e.g., `pgvector`). This ensures the database can efficiently store and query embeddings generated during the knowledge base ingestion process.

```
QUERY 2: Check bedrock_integration schema tables
```

```
Query:  
SELECT  
    table_schema || '.' || table_name as show_tables,  
    table_type  
FROM  
    information_schema.tables  
WHERE  
    table_type = 'BASE TABLE'  
AND  
    table_schema = 'bedrock_integration';
```

show_tables	table_type
bedrock_integration.bedrock_kb	BASE TABLE
bedrock_integration.documents	BASE TABLE
bedrock_integration.knowledge_entries	BASE TABLE
bedrock_integration.vector_metadata	BASE TABLE

(4 rows)

```

Query: SELECT version();

version
PostgreSQL 15.4 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 7.3.0, 64-bit
(1 row)

=====
 QUERY 1: SELECT * FROM pg_extension;
=====

 extname | extowner | extnamespace | extrelocatable | extversion
-----+-----+-----+-----+-----+
 plpgsql | 10 | 11 | f | 1.0
 vector | 16384 | 2200 | t | 0.5.1
 uuid-ossp | 16384 | 2200 | t | 1.1
 pg_stat_statements | 10 | 11 | t | 1.10
(4 rows)

```

Knowledge Base Deployment and Data Sync

Knowledge base successfully deployed

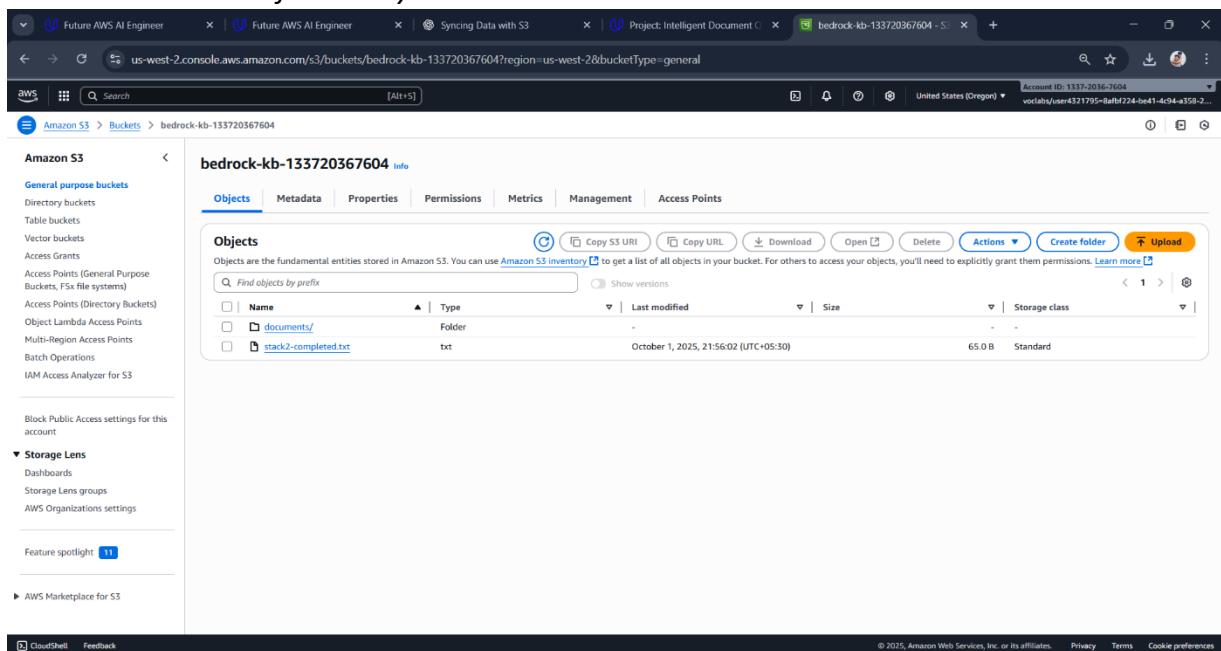
Using Terraform (Stack 2), I deployed the Amazon Bedrock Knowledge Base and connected it with the Aurora vector store. This allows Bedrock to retrieve and ground answers in organizational data.

Criteria: Data from S3 bucket correctly synchronized

Explanation:

I uploaded the project documents (e.g., machine_files.pdf) into the **S3 bucket** using the script `scripts/upload_to_s3.py`. Then, I performed a **data sync operation** from the AWS Console. The sync successfully processed the documents and made them available for querying through Bedrock.

Screenshot: Successful data sync in AWS Console



Objects (5)

Name	Type	Last modified	Size	Storage class
bulldozer-bd850-spec-sheet.pdf	pdf	October 1, 2025, 21:56:04 (UTC+05:30)	448.7 KB	Standard
dump-truck-dt1000-spec-sheet.pdf	pdf	October 1, 2025, 21:56:04 (UTC+05:30)	357.9 KB	Standard
excavator-x950-spec-sheet.pdf	pdf	October 1, 2025, 21:56:04 (UTC+05:30)	274.3 KB	Standard
forklift-fl250-spec-sheet.pdf	pdf	October 1, 2025, 21:56:04 (UTC+05:30)	226.9 KB	Standard
mobile-crane-mc750-spec-sheet.pdf	pdf	October 1, 2025, 21:56:04 (UTC+05:30)	357.7 KB	Standard

3. Python Integration with Bedrock

Criteria: Python function implemented to query the knowledge base

Explanation:

Implemented `query_knowledge_base()` in `bedrock_utils.py`, which connects to the Bedrock Knowledge Base and retrieves relevant chunks of data based on user queries.

Criteria: Successful invocation of the model in bedrock_utils.py

Explanation:

Created the generate_response() function to send retrieved chunks from the knowledge base to Bedrock's LLM. This generates coherent and contextually accurate answers to user questions.

Criteria: Correct implementation of valid_prompt function

Explanation:

Implemented valid_prompt() to categorize and validate prompts. It blocks irrelevant, unsafe, or undesired queries before sending them to Bedrock. This ensures reliability and ethical use of the system.

Criteria: Clear explanation of how temperature and top_p affect responses

Explanation:

- **Temperature** controls randomness:
 - Low values (0.2–0.3) → factual, consistent answers.
 - High values (0.7–0.9) → creative, varied answers.
- **Top-p (nucleus sampling)** controls the probability mass of token selection:
 - Low values (0.3–0.5) → focused, deterministic output.
 - High values (0.9–1.0) → diverse, exploratory output.

For this project, I used **low temperature (0.2)** and **low top-p (0.5)** for factual documentbased answers.

Final output:

The screenshot shows a Streamlit application running at `localhost:8503`. The left sidebar contains configuration sections for AWS Bedrock and Claude 3 Model Selection, including a knowledge base ID and model parameters like Temperature and Top_P. The main area displays a query for the DT1000 Dump Truck, showing 3 relevant documents found. A detailed technical response is provided, mentioning its use in mining operations and its robust construction.

AWS Bedrock Configuration

Knowledge Base Setup:

- Knowledge Base ID: L3CRTSQ79H
- Using KB: L3CRTSQ7...

Claude 3 Model Selection:

- Select LLM Model: anthropic.claude-3...

Model Parameters:

- Temperature: 1.0
- Top_P: 1.0

System Status:

Querying Knowledge Base: L3CRTSQ79H

Found 3 relevant document(s)

DT1000 Dump Truck

Here is a detailed, technical response about the DT1000 Dump Truck based on the provided documentation:

The DT1000 is a large mining dump truck designed for high-productivity, heavy-duty hauling in large-scale open-pit mining operations. It features a robust construction and advanced systems that allow it to excel in extreme conditions, from high temperatures to high altitudes.

Key Specifications:

- Gross Vehicle Weight: 623,690 kg / 1,375,000 lb
- Payload Capacity: 363 metric tons / 400 short tons
- Engine Power: 2,610 kW / 3,500 hp

The DT1000 is engineered around a proven design philosophy focused on five main areas:

- Maximizing payload capacity and efficiency
- Ensuring reliability and durability in extreme conditions

What would you like to know? ➤