nextwork.org

Interacting With My RAG Chatbot in the Terminal

DI Dineshraj Dhanapathy

DI
Dineshraj Dhanapathy
NextWork Student

Introducing Today's Project!

In this project, I will build a chatbot using AWS tools. I'm doing this project to learn how to store data in Amazon S3, create a knowledge base using Amazon Bedrock, and use AI models to enable chatbot conversations. I'll also use AWS CloudShell to interact with the chatbot through the terminal. This hands-on project helps me understand the process of deploying and testing AI applications using real-world AWS services.

Tools and concepts

The services I used were Amazon S3, Amazon Bedrock, and AWS CloudShell. The key concepts I learnt included storing and organizing data in S3, creating a Knowledge Base in Bedrock to connect documents to an AI model, and using the AWS CLI in CloudShell to run commands and test chatbot responses. I also understood how synchronization prepares documents for retrieval and how APIs and AI models work together to power intelligent chat interactions.

Project reflection

This project took me approximately 2 hours to complete. The most challenging part was troubleshooting CLI errors and ensuring the correct Knowledge Base ID and Model ARN were used. It was most rewarding to see the chatbot successfully retrieve information from my documents and respond accurately, proving that all services were connected and working together as expected.

DI

Dineshraj Dhanapathy
NextWork Student

nextwork.org

I did this project today to deepen my understanding of how to build and connect AI chatbots using AWS services. I wanted hands-on experience with S3, Bedrock, and the AWS CLI. This project met my goals by helping me learn how to store data, create a Knowledge Base, and interact with AI models in a real-world workflow. It gave me practical skills I can apply to future AI or cloud-based projects and improved my confidence in using AWS tools effectively.

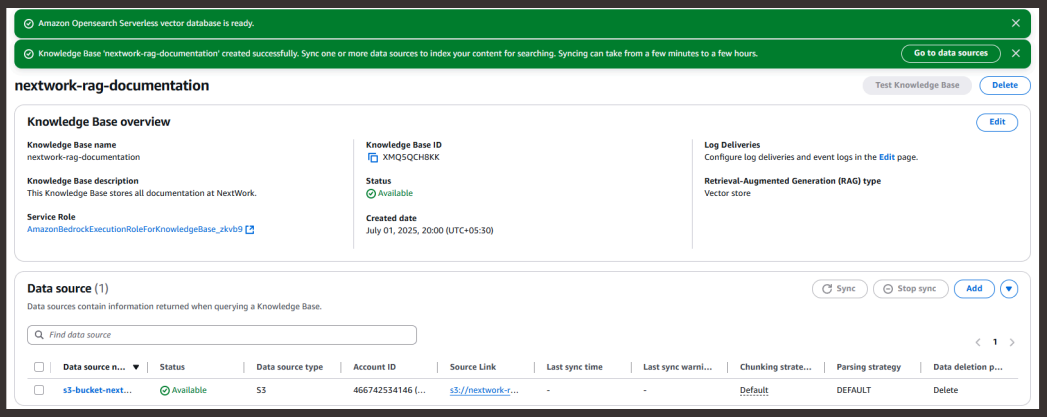
DI
Dineshraj Dhanapathy
NextWork Student

Setting Up The Knowledge Base

To set up my Knowledge Base, I used S3 to store and organize the documents that my chatbot will learn from. The documents I uploaded contain information about a student's portfolio of NextWork project documentation, including their experiences, skills, and achievements. Using S3 allows me to securely store this data in a structured way, making it easy to connect with Amazon Bedrock for further processing. This ensures that the chatbot has accurate and relevant reference material to provide helpful, informed responses.

I also created a Knowledge Base in Bedrock to help my chatbot understand and retrieve information from the documents stored in S3. This allows the chatbot to answer questions using real data instead of just general knowledge. By linking the documents to Bedrock, I make it possible for the AI to search, find, and respond with accurate, context-based answers, making the chatbot more intelligent and useful.

My chatbot also needs access to two AI models, which were selected to generate accurate and helpful responses. I then synchronized the Knowledge Base to ensure it could read and understand the documents stored in S3. This synchronisation processes the content, converts it into a format that the AI models can use, and stores it in the OpenSearch vector database. This step is essential so the chatbot can retrieve relevant information and provide meaningful answers based on the uploaded documents.


Amazon Opensearch Serverless vector database is ready.
Knowledge Base 'nextwork-rag-documentation' created successfully. Sync one or more data sources to index your content for searching. Syncing can take from a few minutes to a few hours.
Go to data sources
nextwork-rag-documentation
Test Knowledge Base
Delete
Knowledge Base overview
Edit
Knowledge Base name
nextwork-rag-documentation
Knowledge Base description
This Knowledge Base stores all documentation at NextWork.
Service Role
AmazonBedrockExecutionRoleForKnowledgeBase_zkvb9
Knowledge Base ID
XMQ5QCH8KK
Status
Available
Created date
July 01, 2025, 20:00 (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.
Sync
Stop sync
Add
Find data source
Data source n...
Status
Data source type
Account ID
Source Link
Last sync time
Last sync warni...
Chunking strate...
Parsing strategy
Data deletion p...
s3-bucket-next...
Available
S3
466742534146 (...
s3://nextwork-r...
-
-
Default
DEFAULT
Delete

Running CLI Commands in CloudShell

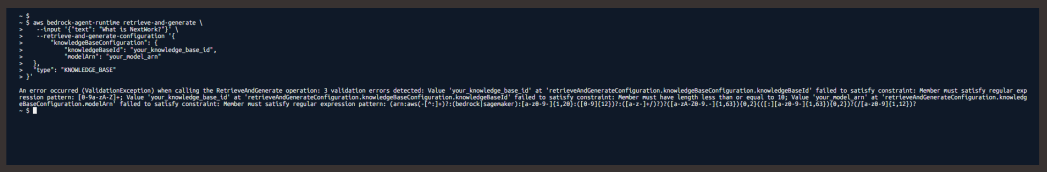
AWS CLI is a command line tool that lets me create, manage, and interact with AWS services using simple text commands instead of clicking through the AWS Management Console. To start testing CLI commands, I first opened CloudShell, which is a browser-based shell environment built into the AWS Console. CloudShell comes with AWS CLI already installed, so I don't need to set anything up on my computer. It makes it easy and fast to run commands, test my chatbot, and manage AWS resources directly.

When I first ran a Bedrock command, I ran into an error because I needed to provide values for the knowledgeBaseId and the modelArn. The error message showed that placeholders like 'your_knowledge_base_id' and 'your_model_arn' were not replaced with real values. AWS CLI requires these values to match specific patterns and length constraints. To fix this, I looked up the actual Knowledge Base ID and the correct model ARN from the AWS Console and replaced the placeholders in the command. Once corrected, the command ran successfully.

While finding the parameters takes extra time, the advantage of using the CLI is that it allows you to manage AWS resources more quickly and efficiently through simple commands. It's especially useful for automation, scripting, and handling repetitive tasks. The CLI also gives you more control and flexibility compared to the AWS Console, making it ideal for experienced users who want to work faster and customize their workflows with precision.

DI

Dineshraj Dhanapathy
NextWork Student



Running Bedrock Commands

To find the required values, I had to go to the Amazon Bedrock console, open my Knowledge Base, and copy the Knowledge Base ID. Then, I selected the AI model I wanted to use (like Llama 3 70B Instruct or Titan Text Embeddings V2) and copied its Model ARN. I updated my CLI command with these values. The Bedrock command ran successfully and showed me a response with relevant information retrieved from my S3 documents, including details from files like Prompt Engineering.pdf and Automate Your Browser with AI Agents.pdf, confirming my chatbot is now working correctly.

The retrieve-and-generate command typically also outputs raw metadata, references, and document source details along with the generated response. To tidy up the terminal response, I added the --query 'output.text' parameter. This filtered the output to show only the chatbot's answer, making it cleaner and easier to read. Instead of seeing all the technical details, the terminal now displayed just the final response, like: “NextWork is a platform that offers projects and resources for learning and development.”


CloudShell
Actions
ap-south-1
{
"generatedResponsePart": {
"textResponsePart": {
"span": {
"end": 86,
"start": 0
},
"text": "NextWork is a platform that offers projects and resources for learning and development."
}
},
"retrievedReferences": [
{
"content": {
"text": "Our prompt template will focus on order tracking. Our initial prompt template identified our stakeholder's needs by discussing current issues and solutions. Our initial prompt template includes the desired tone, steps to resolve the issue, background con
text, escalation criteria and example scripts The feedback we received was to include more edge cases like the customer not having an order ID, or wanting refund. To respond to feedback, we updated our template to include the mentioned edge cases, and including an example successful con
versation https://community.nextwork.org/c/i-have-a-question?automatic_login=trueNextWork.org Everyone should be in a job they love. Check out nextwork.org for more projects https://community.nextwork.org/c/i-have-a-question?automatic_login=true https://commu
nity.nextwork.org/c/i-have-a-question?automatic_login=true",
"type": "TEXT"
},
"location": {
"s3Location": {
"uri": "s3://nextwork-rag-documentation-dd/Prompt Engineering.pdf"
},
"type": "S3"
},
"metadata": {
"x-amz-bedrock-kb-source-uri": "s3://nextwork-rag-documentation-dd/Prompt Engineering.pdf",
"x-amz-bedrock-kb-document-page-number": 8.0,
"x-amz-bedrock-kb-chunk-id": "1%3A0%3Ak1eJxpcBEHtXhn6J17Vv",
"x-amz-bedrock-kb-data-source-id": "GBRWEBVKQV"
}
},
{
"content": {
"text": "https://community.nextwork.org/c/i-have-a-question?automatic_login=trueNatasha Ong NextWork Student NextWork.org Advanced Navigation We also tried telling the agent to check out a NextWork PRO project and see what happens when it tries to start the p
roject. It reported that it requires a PRO subscription/PRO access in order to start. https://community.nextwork.org/c/i-have-a-question?automatic_login=trueNatasha Ong NextWork Student NextWork.org Debugging and Logs When our agent encounters an error, three ways we can d
ebug it are: finding out whether automated access to the website is blocked; be clearer about the format of our desired result; give more specific results if the website structure is complex. The terminal logs show us a breakdown of our agent's thoughts, plans and memory with every sing
le step as it tried to complete the task. This helps us understand how the AI model thinks/analyses a page and where it might've gotten stuck. https://community.nextwork.org/c/i-have-a-question?automatic_login=trueNatasha Ong NextWork Student NextWork.org Advanced Browser Conf
iguration In this project extension, we will set up WebUI agents to control our own browser - i.e. Google Chrome with our existing login details, instead of the default Chromium browser. Using your own browser is useful for accessing websites with logins.",
"type": "TEXT"
},
"location": {
"s3Location": {
"uri": "s3://nextwork-rag-documentation-dd/Automate Your Browser with AI Agents.pdf"
},
"type": "S3"
},
"metadata": {
"x-amz-bedrock-kb-source-uri": "s3://nextwork-rag-documentation-dd/Automate Your Browser with AI Agents.pdf",
"x-amz-bedrock-kb-document-page-number": 5.0,
"x-amz-bedrock-kb-chunk-id": "1%3A0%3AgUeiHpcBlPJ9T2PVuRb_",
"x-amz-bedrock-kb-data-source-id": "GBRWEBVKQV"
}
}
]
}

DI
Dineshraj Dhanapathy
NextWork Student
nextwork.org

Extending Your Knowledge Base

In a project extension, I asked my Knowledge Base about a text—"Why did the coffee go to the police?" I noticed that the response was: “Sorry, I am unable to assist you with this request.” This showed that while the Knowledge Base is working technically, it may not respond to queries outside the context of the uploaded documents. It reminded me that the chatbot is designed to answer based on specific content and may not handle unrelated questions.

To add new information to my Knowledge Base, I ran commands to upload new documents to my S3 bucket and then triggered a sync by running the appropriate CLI commands with my Knowledge Base ID, data source ID, and ingestion job ID. This ensured the new content was processed and indexed for retrieval. Compared to using the console, this process was quicker and more efficient, especially when managing updates frequently, as it allowed me to automate and monitor everything directly from the terminal.

To validate the update worked, I ran a retrieve-and-generate command using AWS CLI with a new query. The Knowledge Base successfully responded with the updated information, showing that it could access and retrieve data from the newly uploaded documents. For example, when I asked, “Why did the coffee go to the police?” it correctly responded with “Because it got mugged!”—confirming that the synchronization included recent content and the chatbot was functioning as expected.

```
~ $ aws bedrock-agent get-ingestion-job \
>     --knowledge-base-id "XMQ5QCH8KK" \
>     --data-source-id "GBRWEBV9QV" \
>     --ingestion-job-id "PQOJGFRQU9"
{
    "ingestionJob": {
        "dataSourceId": "GBRWEBV9QV",
        "ingestionJobId": "PQOJGFRQU9",
        "knowledgeBaseId": "XMQ5QCH8KK",
        "startedAt": "2025-07-01T16:31:14.899934+00:00",
        "statistics": {
            "numberOfDocumentsDeleted": 0,
            "numberOfDocumentsFailed": 0,
            "numberOfDocumentsScanned": 11,
            "numberOfMetadataDocumentsModified": 0,
            "numberOfMetadataDocumentsScanned": 0,
            "numberOfModifiedDocumentsIndexed": 0,
            "numberOfNewDocumentsIndexed": 1
        },
        "status": "COMPLETE",
        "updatedAt": "2025-07-01T16:31:17.994619+00:00"
    }
}
~ $
```

DI
Dineshraj Dhanapathy
NextWork Student

Interacting with AI Models Directly

On top of chatting with my chatbot, I also interacted directly with an AI model via the terminal by running a retrieve-and-generate command using the AWS CLI. This allowed me to send a prompt directly to the model and receive a generated response, like a short poem about a dog. It demonstrated that I could bypass the Knowledge Base if needed and engage with the model for creative or general-purpose tasks. This direct interaction was fast, flexible, and showed the model's standalone capabilities.