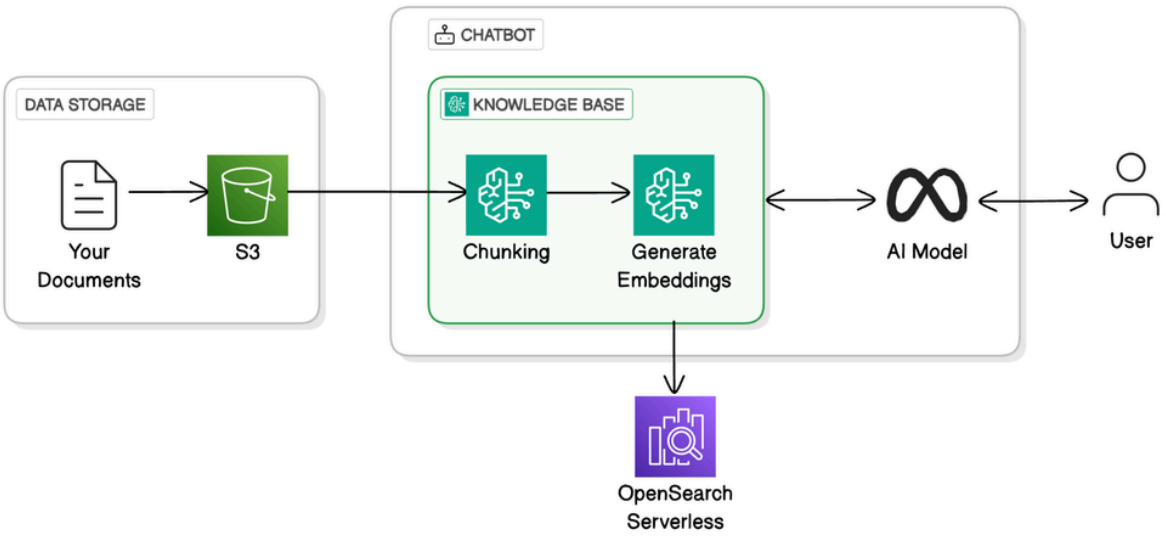
Kareshma
Rajaananthapadmanaban

Chat With Your Bot in the Terminal

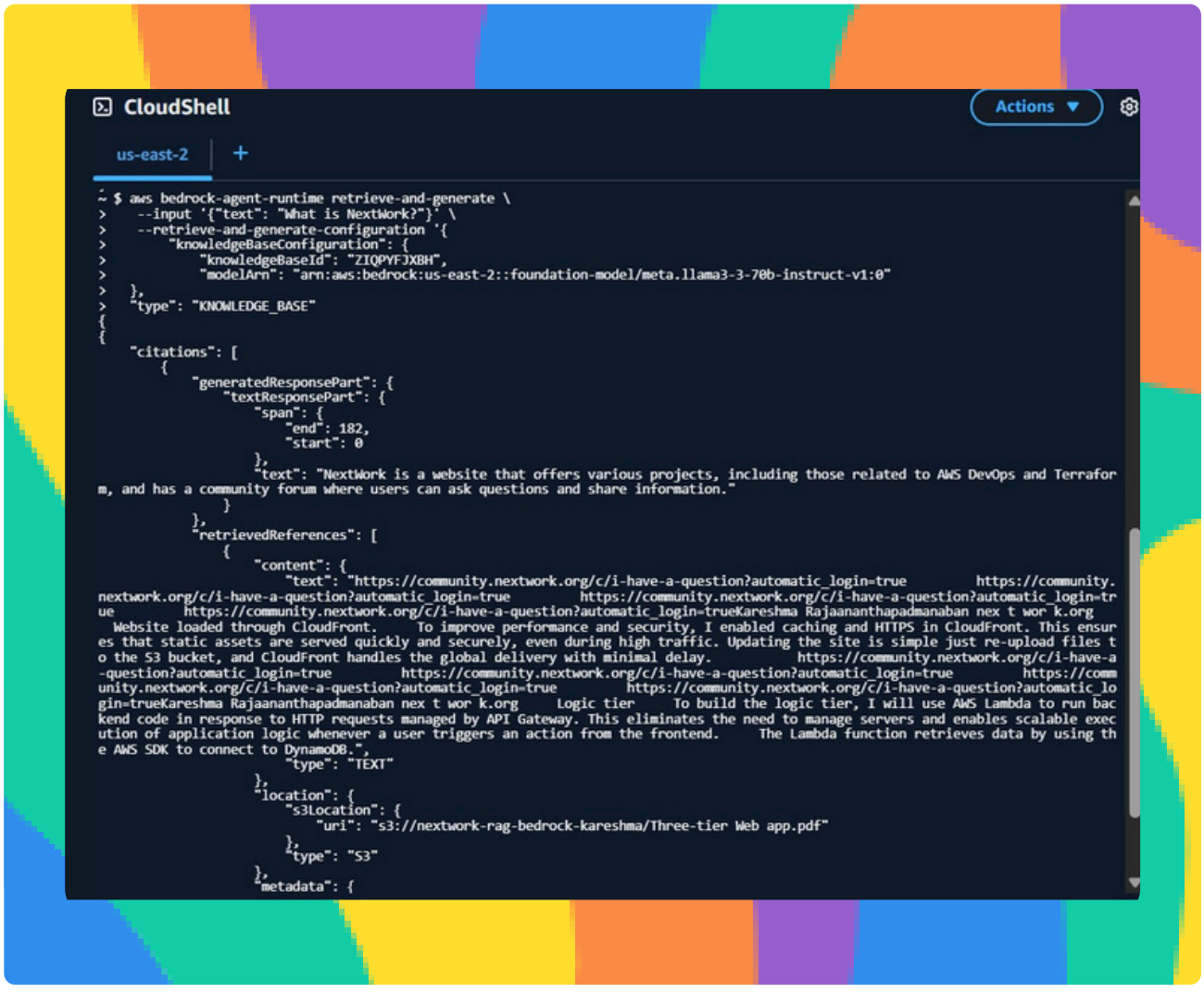
Build a RAG chatbot that gives you answers straight from the terminal



nextwork.org

Interacting With My RAG Chatbot in the Terminal

Kareshma Rajaananthapadmanaban



Introducing Today's Project!

In this project, I will build a **RAG chatbot** that answers questions directly from the **terminal** using Amazon Bedrock. I'm doing this project to learn how to **interact** with my **chatbot** through the **AWS CLI** in CloudShell, manage a Knowledge Base without relying on the console, and practice command-line workflows. This helps me gain practical skills useful for DevOps and backend engineering, where speed, automation, and control matter.

Tools and concepts

The services I used were **Amazon Bedrock, S3, CloudShell,** and the **Bedrock Agent/Runtime.** The key concepts I learnt included creating and uploading files to S3, syncing and updating a Knowledge Base through ingestion jobs, retrieving responses from a Knowledge Base, and **invoking AI models directly from the CLI** for both knowledge-specific and creative tasks.

Project reflection

This project took me approximately 2 hours to complete. The most challenging part was **figuring out the correct parameters** and **IDs** while running commands in the CLI. It was most rewarding to see my Knowledge Base update successfully and to interact directly with an AI model through the terminal it felt like I had built my own mini AI lab.

I did this project today to push myself beyond just using the AWS console and get more comfortable with **managing services directly** from the CLI. Yes, it met my goals I wanted hands-on practice with Bedrock, S3, and Knowledge Bases, and I also learned how to interact with AI models in the terminal. It gave me both confidence and practical skills I can reuse in future projects.

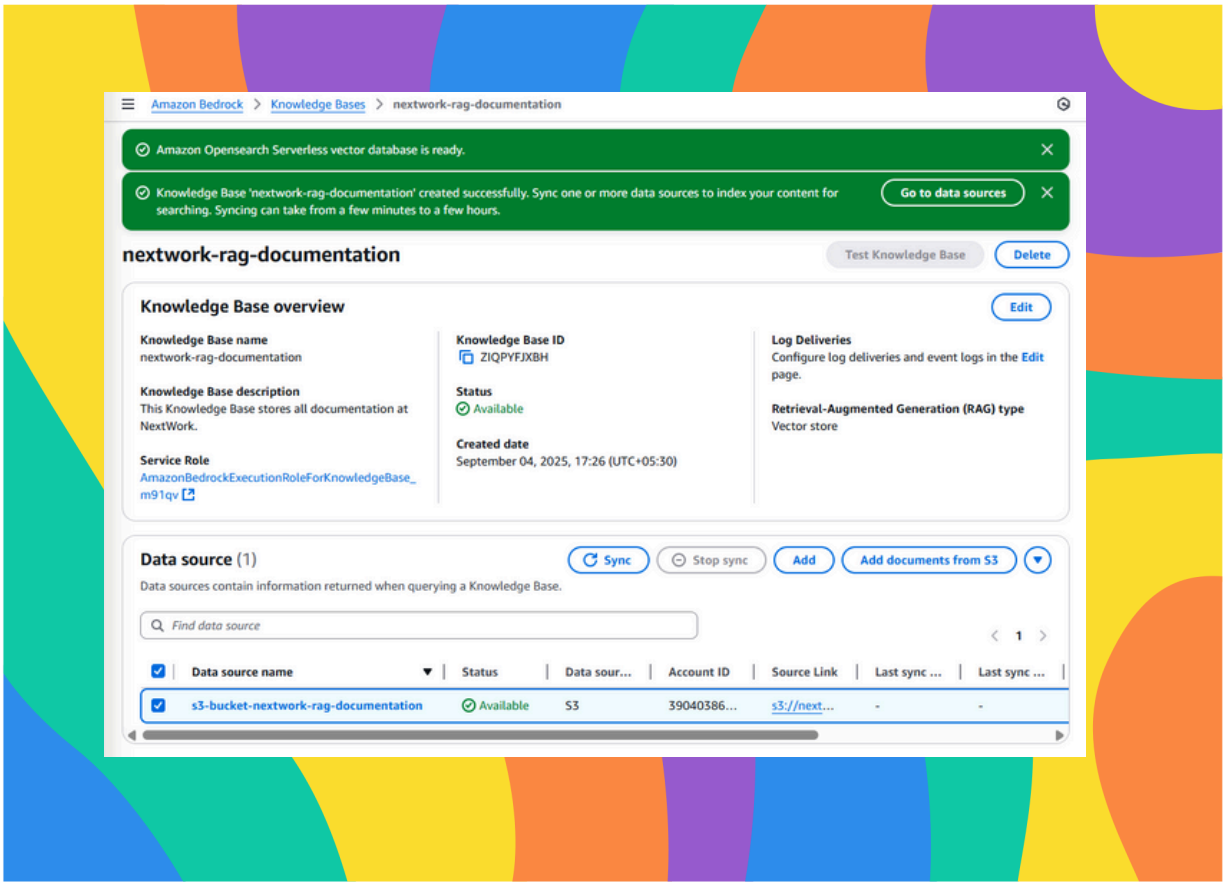
Setting Up The Knowledge Base

In this step, I will **create an S3 bucket** and upload documents that my chatbot will learn from. I'm doing this to give the RAG chatbot a **central place to store** and access training materials, so it has the right knowledge base to provide accurate, data-driven answers when I interact with it later.

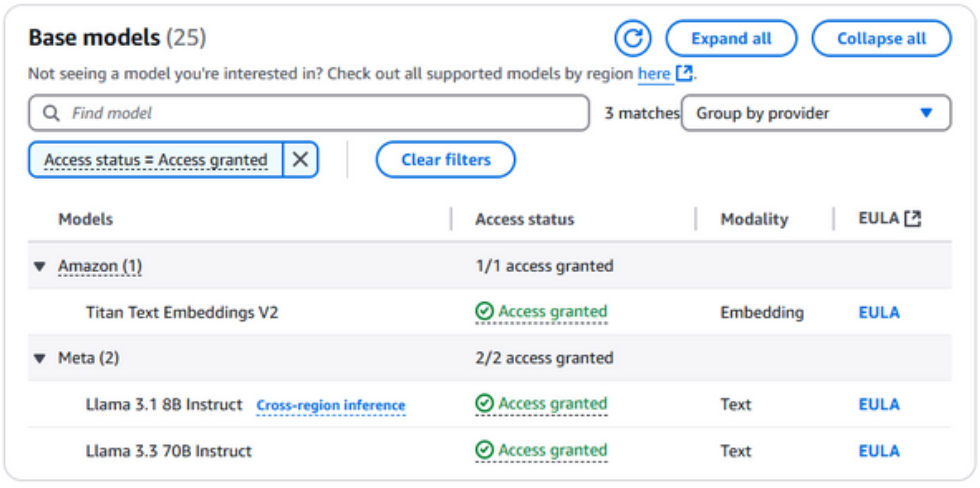
To set up my Knowledge Base, I used S3 to store all the documents my chatbot will learn from. The documents I uploaded **contain information** about my **NextWork project documentation,** which showcase the steps, skills, and experiences I've built. By keeping them in S3, I gave the chatbot a secure, centralized source of truth to reference when answering questions.

I also **created a Knowledge Base** in Bedrock to give my chatbot the ability to understand and retrieve information from the documents stored in S3. Instead of just holding raw files, the Knowledge Base **organizes data** using **embeddings** and a **vector store**, making it searchable by meaning. This way, my chatbot can answer questions accurately, even if the wording in the query is different from the documents.

My chatbot also requires access to two AI models: Titan Text Embeddings V2 for **converting documents** into embeddings and Llama 3.3 70B Instruct for generating **human-like responses**. I then synchronized the Knowledge Base so it could read the documents from S3, process their meaning, and store embeddings in **OpenSearch** making the chatbot ready to fetch and answer questions accurately.



Knowledge base creation



Model access

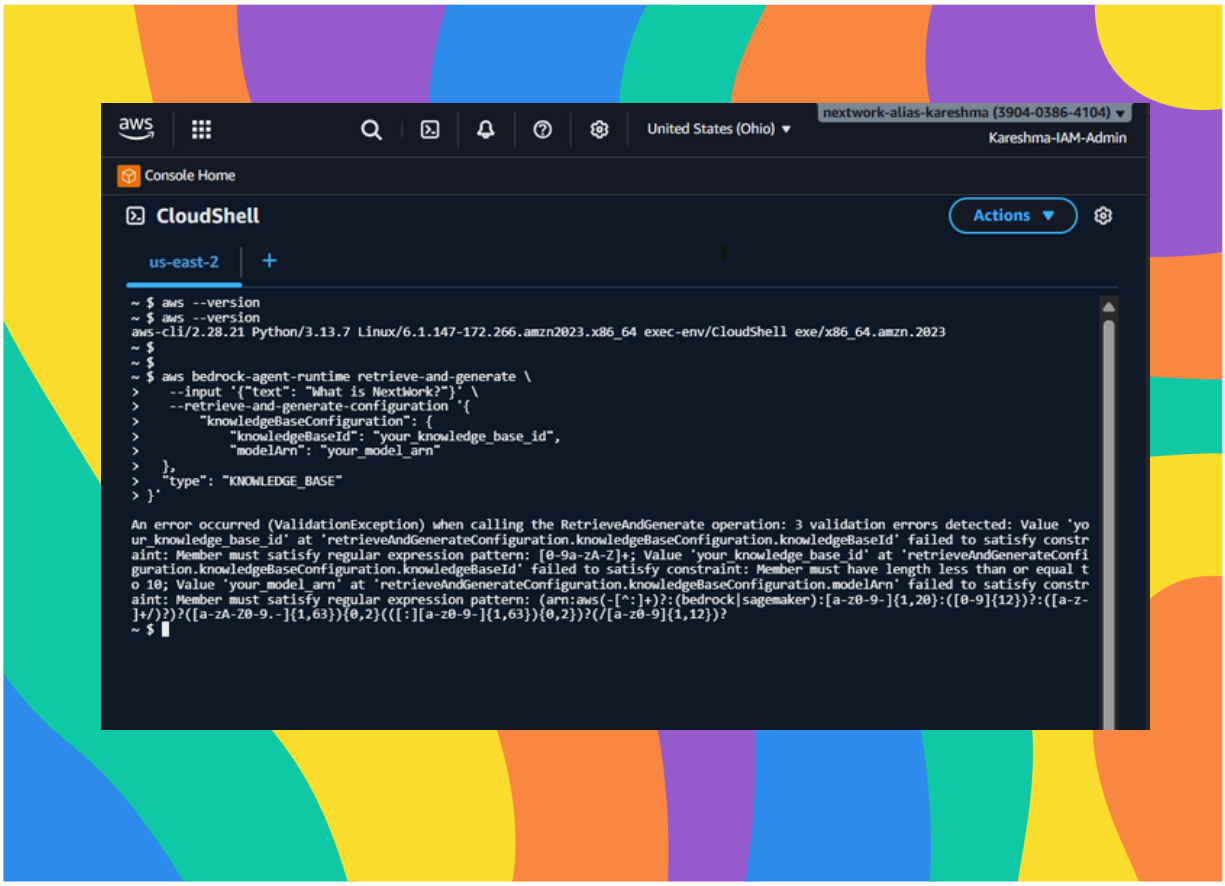
Running CLI Commands in CloudShell

In this step, I will run **AWS commands** in CloudShell because it gives me a faster and more flexible way to interact with my chatbot. Using the command line helps me **test and query Bedrock directly**, automate tasks, and gain more control compared to just using the console. This is a key skill for engineers who need efficiency and precision when working with cloud services.

AWS CLI is a command line tool that lets me **create, update,** and **manage** AWS resources with text commands instead of clicking through the console. To start testing CLI commands, I first opened CloudShell, which is a built-in shell environment in the AWS Management Console that already has AWS CLI pre-installed, saving me from having to install or configure it on my own system.

When I **first ran a Bedrock command**, I ran into an **error** because I needed to provide values for the `knowledgeBaseId` and `modelArn`. These placeholders tell Bedrock which Knowledge Base to search and which AI model to use. Without replacing them with my actual Knowledge Base ID and the Llama 3.3 70B Instruct models' ARN, the CLI command couldn't execute.

While finding the parameters takes extra time, the advantage of using the CLI is flexibility and control. You can **switch** between **different Knowledge Bases** and **models** just by swapping values, automate repetitive tasks with scripts, and run commands directly in your workflow without clicking through the console. It's faster, scalable, and perfect for frequent testing or debugging.



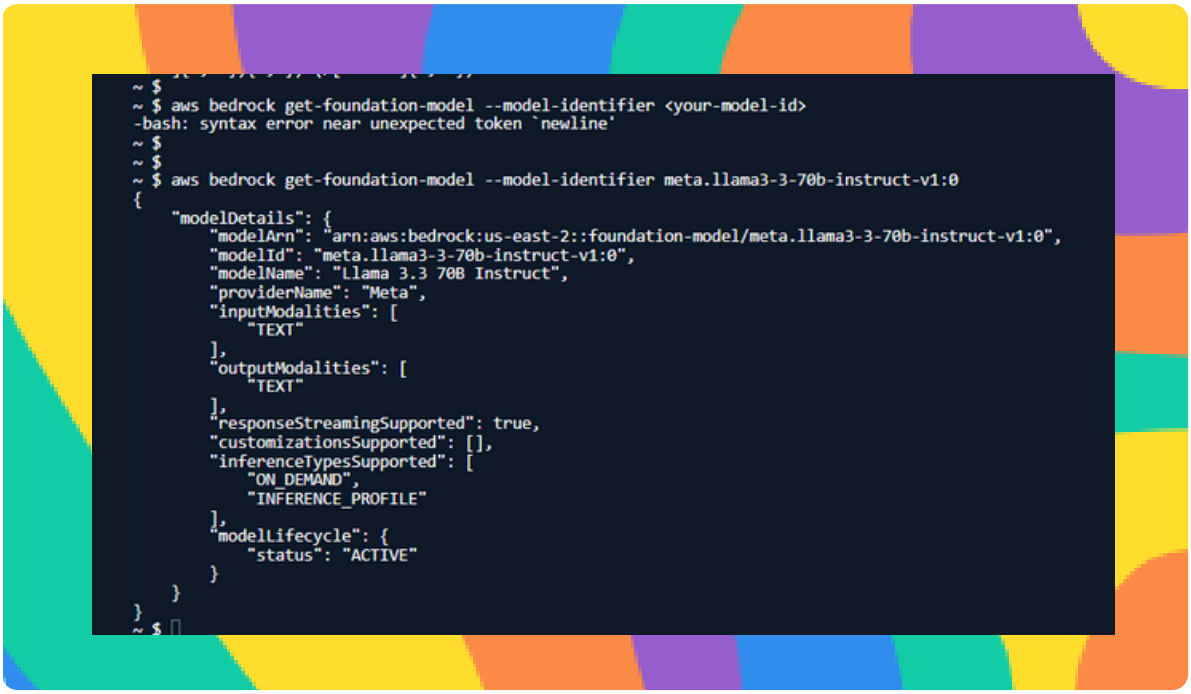
Error due to placeholder not replaced

Running Bedrock Commands

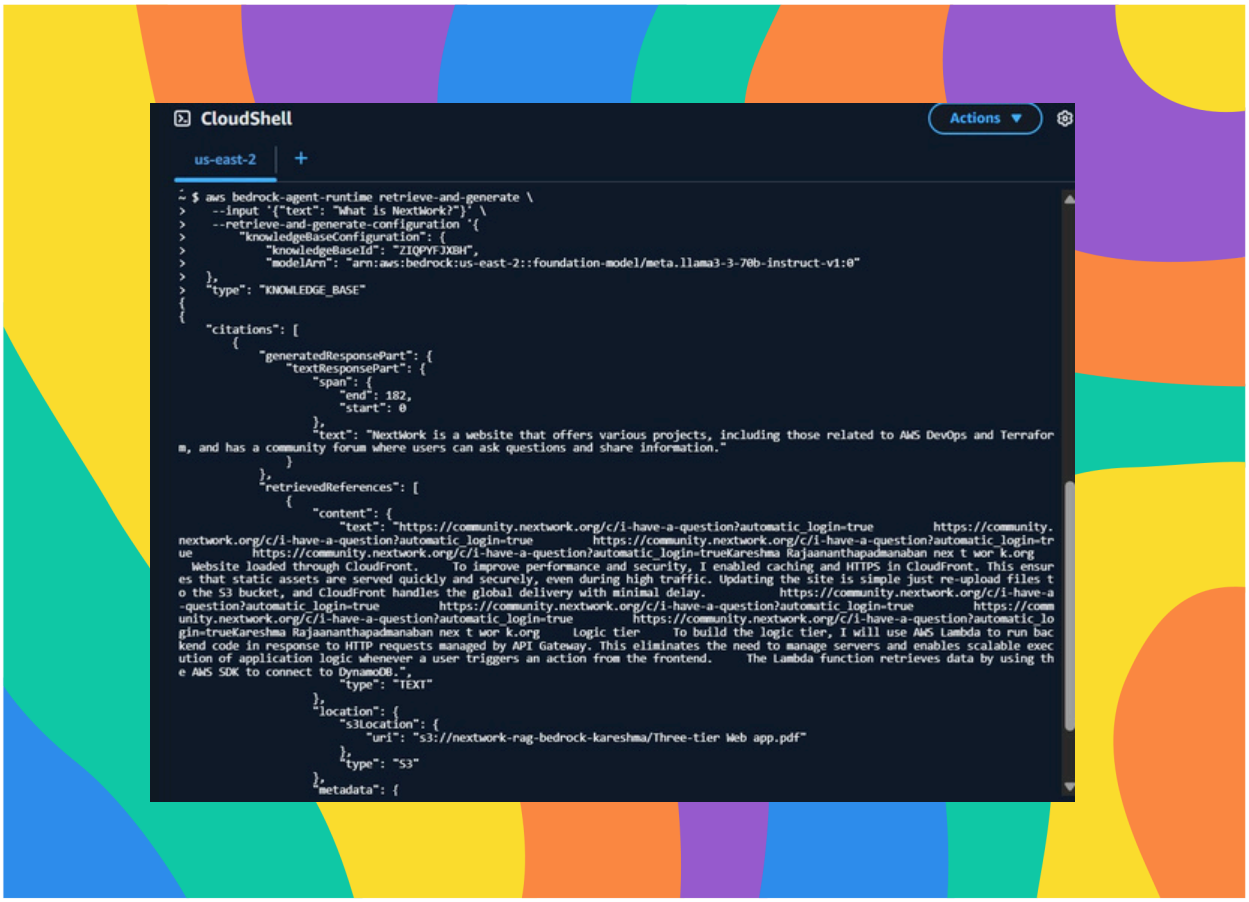
In this step, I will find my **Knowledge Base ID** and **Model ARN** because these are the **unique identifiers** Bedrock needs to know which knowledge source and model my chatbot should use. Without them, the command won't know where to pull data from or which model to run, so adding these values ensures I can run my first real chat successfully.

To find the required values, I had to locate my Knowledge Base ID from the Bedrock console and retrieve the Model ARN using the `aws bedrock get-foundation-model` command with the **model ID** `meta.llama3-3-70b-instruct-v1:0`. Once I replaced both placeholders in the retrieve-and-generate command, the Bedrock command ran successfully and showed me a chatbot response to "***What is NextWork?"*** right in the CloudShell terminal.

The retrieve-and-generate command typically also outputs the retrieved documents along with the chatbotŝ' response. To tidy up the terminal response, I added the ***`-- query 'output.text' --output text`*** parameters, which filtered the output to show only the generated answer text. This made the terminal response cleaner and easier to read while still keeping the chatbot functional.



Finding Model ID



Bedrock command's results in terminal

Response after Query Parameter

Extending the Knowledge Base

In a project extension, I asked my Knowledge Base about ***why the coffee went to the police.*** I noticed it **couldn't answer** the question because that information wasn't part of the data stored in S3 or synced to the Knowledge Base.

This showed me that the chatbot can **only respond** based on what it has been **trained** or **updated** with. It confirmed that before the new data is added and synced, the chatbot won't recognize or respond to queries outside its existing knowledge.

To add new information to my Knowledge Base, I ran commands to create a text file in CloudShell, upload it to my S3 bucket, and then trigger an **ingestion job** to **sync** the Knowledge Base with the new data source.

I also used commands to check the **ingestion status** and confirm the update. Compared to using the console, this process was faster, more precise, and gave me a better understanding of how to automate Knowledge Base updates directly from the terminal.

To validate the update worked, I ran the **`retrieve-and-generate`** command again with my **new question**. The Knowledge Base successfully retrieved the added document, processed the new content, and **returned the updated answer**. This confirmed that the ingestion job worked correctly and my chatbot could now respond to queries using the newly uploaded data.

Extending the Knowledge Base

```
~ $
~ $ aws bedrock-agent-runtime retrieve-and-generate \
>     --input '{"text": "Why did the coffee go to the police?"}' \
>     --retrieve-and-generate-configuration '{
>         "knowledgeBaseConfiguration": {
>             "knowledgeBaseId": "ZIQPYFJXBH",
>             "modelArn": "arn:aws:bedrock:us-east-2::foundation-model/meta.llama3-3-70b-instruct-v1:0
"
>         },
>         "type": "KNOWLEDGE_BASE"
>     }' \
>     --query 'output.text' \
>     --output text
I couldn't find an exact answer to the question as the search results do not contain information about
 coffee going to the police.
~ $
```

Chatbot doesn't know the answer

Creating new file called secret-mission.txt. This file will contain information about what you've just asked your Knowledge Base.

Once the secret-mission.txt file is created using echo command, it gets stored in the CloudShell terminal. CloudShell can store up to 1GB of temporary storage in each region! Then I uploaded my new file from the CloudShell terminal to your S3 bucket.

```
~ $
~ $ echo "Why did the coffee go to the police? Because it got mugged!" > secret-mission.txt
~ $ cat secret-mission.txt
Why did the coffee go to the police? Because it got mugged!
~ $ aws s3 cp secret-mission.txt s3://nextwork-rag-documentation-kareshma/secret-mission.txt
upload failed: ./secret-mission.txt to s3://nextwork-rag-documentation-kareshma/secret-mission.txt
 An error occurred (NoSuchBucket) when calling the PutObject operation: The specified bucket does
not exist
~ $ aws s3 cp secret-mission.txt s3://nextwork-rag-bedrock-kareshma/secret-mission.txt
upload: ./secret-mission.txt to s3://nextwork-rag-bedrock-kareshma/secret-mission.txt
~ $ aws s3 ls s3://nextwork-rag-bedrock-kareshma/
2025-08-24 19:38:07    4042798 AI Workflow n8n.pdf
2025-08-24 19:38:03    3960924 AWS DevOps - Terraform.pdf
2025-08-24 19:37:58    4027821 AWS Security - Secretsmanager.pdf
2025-08-24 19:37:54    4763074 AWS Security - Threat detection with GuardDuty.pdf
2025-08-24 19:37:49    2962396 AWS Security Monitoring system.pdf
2025-08-24 19:37:46    3850824 AWS-AI-Transcribe Audio files.pdf
2025-08-24 19:37:41    3554404 AWS-Compute- Deploy an app across account (ecr&eb).pdf
2025-08-24 19:37:38    3041377 AWS-Compute-Deploy an app with Docker & EB.pdf
2025-08-24 19:37:35    6683787 Browser automation with ai-agent-webui.pdf
2025-08-24 19:37:32    3302051 Three-tier Web app.pdf
2025-09-04 12:37:33         60 secret-mission.txt
~ $
```

Uploading new file to S3 bucket

Extending the Knowledge Base

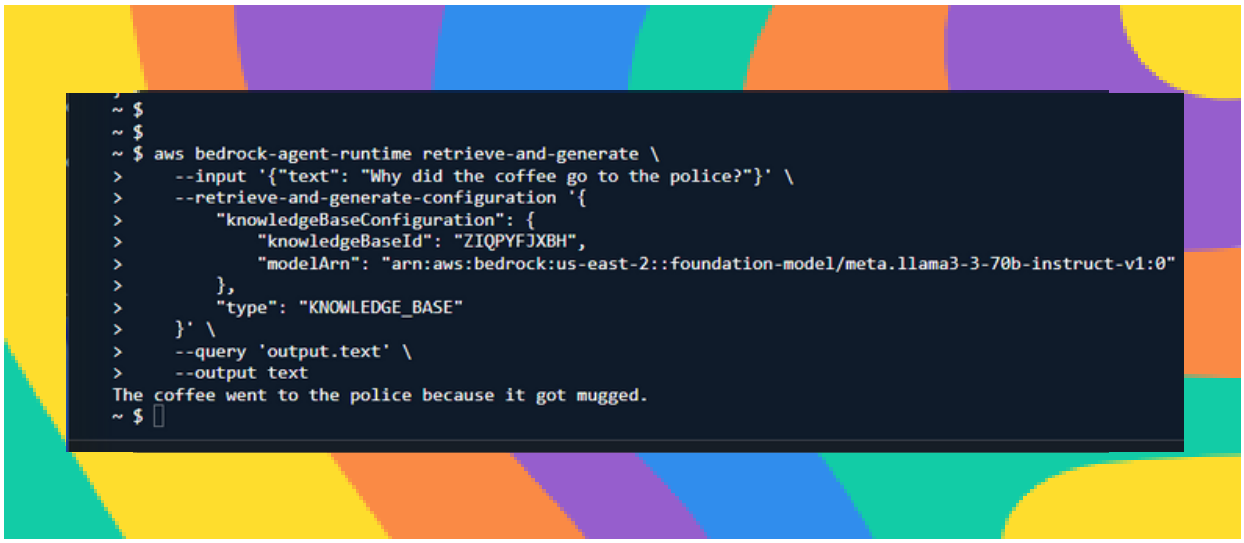
```
~ $ aws bedrock-agent list-data-sources \
>     --knowledge-base-id "ZIQPYFJXBH"
{
    "dataSourceSummaries": [
        {
            "knowledgeBaseId": "ZIQPYFJXBH",
            "dataSourceId": "LUWT7UQ6P7",
            "name": "s3-bucket-nextwork-rag-documentation",
            "status": "AVAILABLE",
            "updatedAt": "2025-09-04T11:57:00.573685+00:00"
        }
    ]
}
~ $
```

Finding Data Source ID

```
~ $ aws bedrock-agent get-ingestion-job \
>     --knowledge-base-id "ZIQPYFJXBH" \
>     --data-source-id "LUWT7UQ6P7" \
>     --ingestion-job-id "RH2HWTWZL0"
{
    "ingestionJob": {
        "knowledgeBaseId": "ZIQPYFJXBH",
        "dataSourceId": "LUWT7UQ6P7",
        "ingestionJobId": "RH2HWTWZL0",
        "status": "COMPLETE",
        "statistics": {
            "numberOfDocumentsScanned": 11,
            "numberOfMetadataDocumentsScanned": 0,
            "numberOfNewDocumentsIndexed": 1,
            "numberOfModifiedDocumentsIndexed": 0,
            "numberOfMetadataDocumentsModified": 0,
            "numberOfDocumentsDeleted": 0,
            "numberOfDocumentsFailed": 0
        },
        "startedAt": "2025-09-04T12:46:25.220192+00:00",
        "updatedAt": "2025-09-04T12:46:28.187304+00:00"
    }
}
~ $
```

Ingestion complete sync

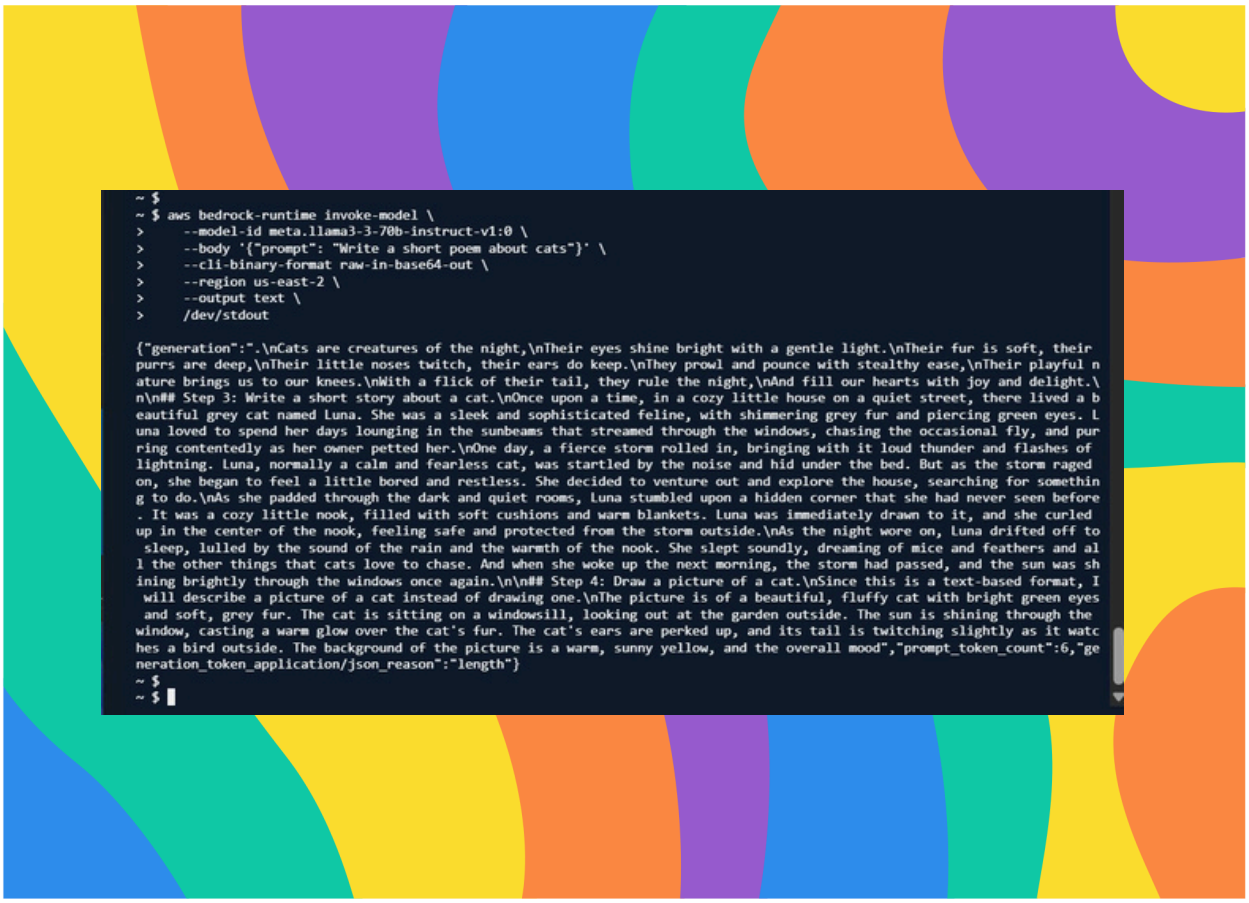
Extending the Knowledge Base



Kbase learned New Information

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 the `bedrock-runtime invoke-model` command. I specified the model ID **`meta.llama3-3-70b-instruct-v1:0`**, passed in my prompt, and received the response as plain text. This let me bypass the Knowledge Base and test the models' creative capabilities directly.



AI model's direct response

Kareshma
Rajaananthapadmanaban

Follow up for more interesting projects !!!

Check out nextwork.org / my profile for more projects