UTD GenAI TA – AI-Powered Learning Assistant for Students

Introduction

"Welcome to UTD GenAI TA – an AI-powered learning assistant designed to help UTD students with their coursework. Instead of relying solely on Teaching Assistants, students can now get instant, accurate responses to their queries based on class materials, lecture slides, and professor-provided documents."

How It Works

1. Uploading Course Materials

"The first step in building this system is creating a knowledge base from course materials. Professors or TAs upload lecture slides, class notes, research papers, and other relevant documents to an S3 bucket in AWS."

2. Creating the Knowledge Base

*"Once the materials are uploaded, we configure a **Knowledge Base** in AWS Bedrock. This knowledge base automatically processes documents, converting them into **vector embeddings** for efficient retrieval. The system also breaks the text into chunks, making it easier to locate relevant information."*

3. AI-Driven Response Generation

*"The Lambda function interacts with the Knowledge Base, retrieving the most relevant information from the uploaded course materials. This retrieved data is then passed to an **AWS Bedrock Foundation Model**, which generates an intelligent, contextual response. The AI doesn't just guess—it bases its answers directly on course content."*

Example Use Cases

📌 **Scenario 1: Understanding Lecture Concepts**
💡 *Student Question: "Can you explain the difference between normalization and denormalization from my database class?"*
🔷 *Response:* The AI retrieves relevant sections from the professor's slides and class notes, providing a precise answer along with references to the original material.

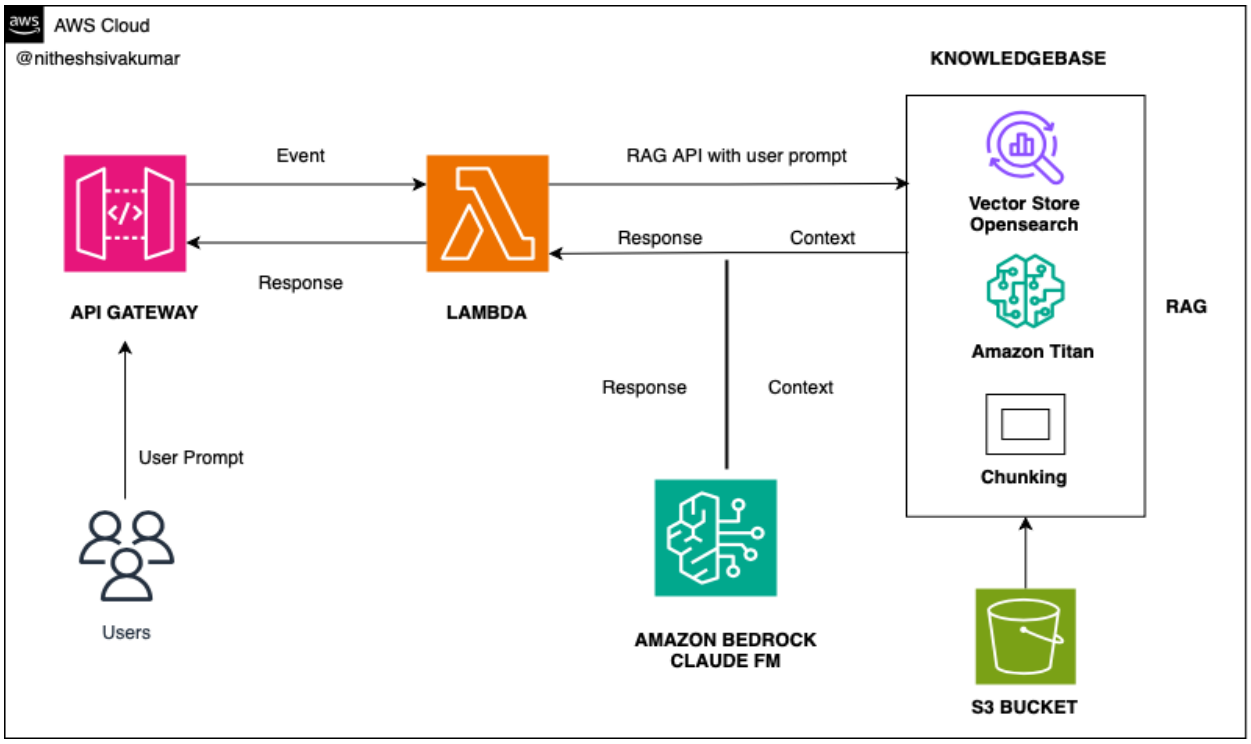
📌 **Scenario 2: Homework Help**
💡 *Student Question: "How do I solve this problem from last week's assignment?"*
🔷 *Response:* The system finds the related example from lecture slides and presents a structured explanation.

📌 **Scenario 3: Exam Preparation**
💡 *Student Question: "What are the key topics covered in last semester's cybersecurity midterm?"*
🔷 *Response:* The AI summarizes key points based on past exam materials.

Testing the System

*"To ensure accuracy, I tested the system using **Postman** by making API requests. Queries related to different topics successfully returned detailed answers, including references to the original documents. The AI also provides **source citations**, so students know exactly where the answer is coming from."*

Architecture:



Architecture Overview

1. **S3 Bucket as a Data Source**
    - A dedicated Amazon S3 bucket is created to store PDF documents related to AWS services, including FAQs on EC2, EBS, and ECS.
2. **Knowledge Base for Retrieval-Augmented Generation (RAG)**
    - A Knowledge Base in AWS Bedrock is configured to process and retrieve relevant information from stored documents.
    - It handles document chunking, vector embedding, and storage in a vector database.
3. **API Gateway for Client Requests**
    - Amazon API Gateway is deployed as the front-facing endpoint for the application, allowing users to send queries.
4. **AWS Lambda for Processing Requests**
    - A Lambda function is implemented to receive user queries from API Gateway.
    - The function invokes the **Retrieve and Generate API** of the Knowledge Base to fetch contextual information.
    - The retrieved context is sent to a Bedrock Foundation Model, which generates a relevant response.
5. **Response Generation & Delivery**
    - The generated answer is returned to AWS Lambda.
    - The Lambda function sends the final response to the user through API Gateway.
    - The system also provides **citations**, showing the source of the retrieved answer.

Workflow Summary

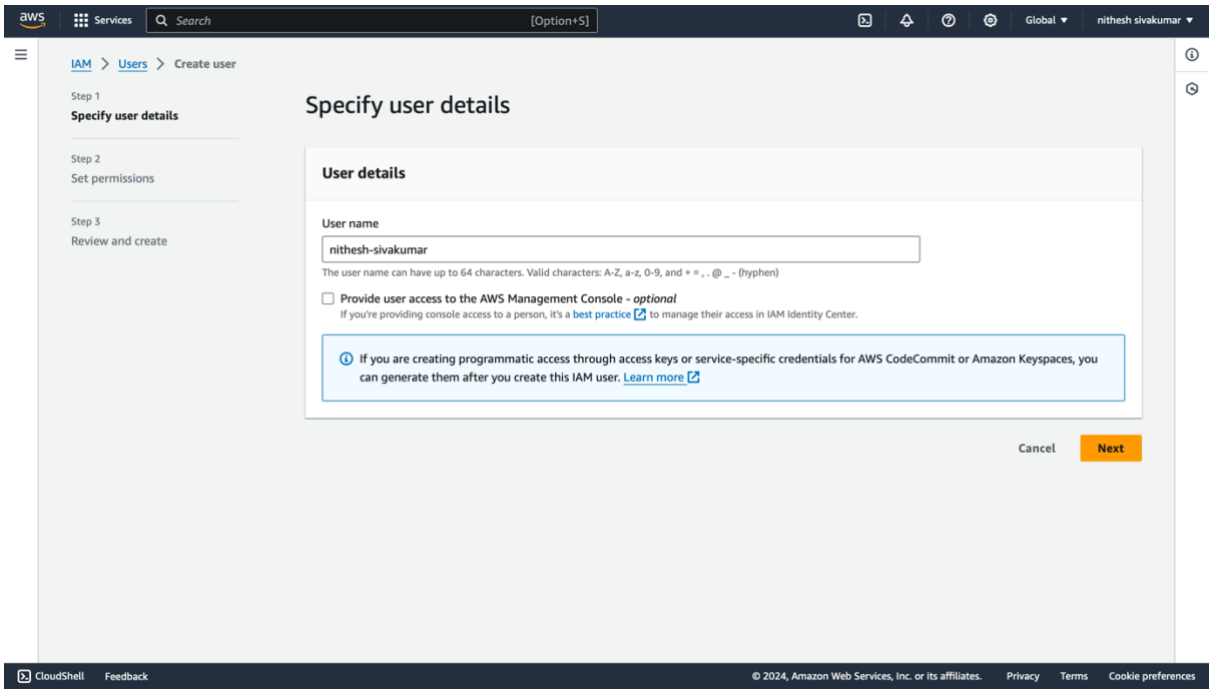
1. A **user** asks a question (e.g., “Which EBS volume is best for high throughput?”).
2. The query is sent to **API Gateway**, which triggers the **Lambda function**.
3. **Lambda** invokes the **Retrieve and Generate API** from the **Knowledge Base**, retrieving relevant data from the S3-stored documents.
4. The **Bedrock Foundation Model** generates an AI-powered response using the retrieved context.
5. The response, along with **source citations**, is returned to the user via **API Gateway**.

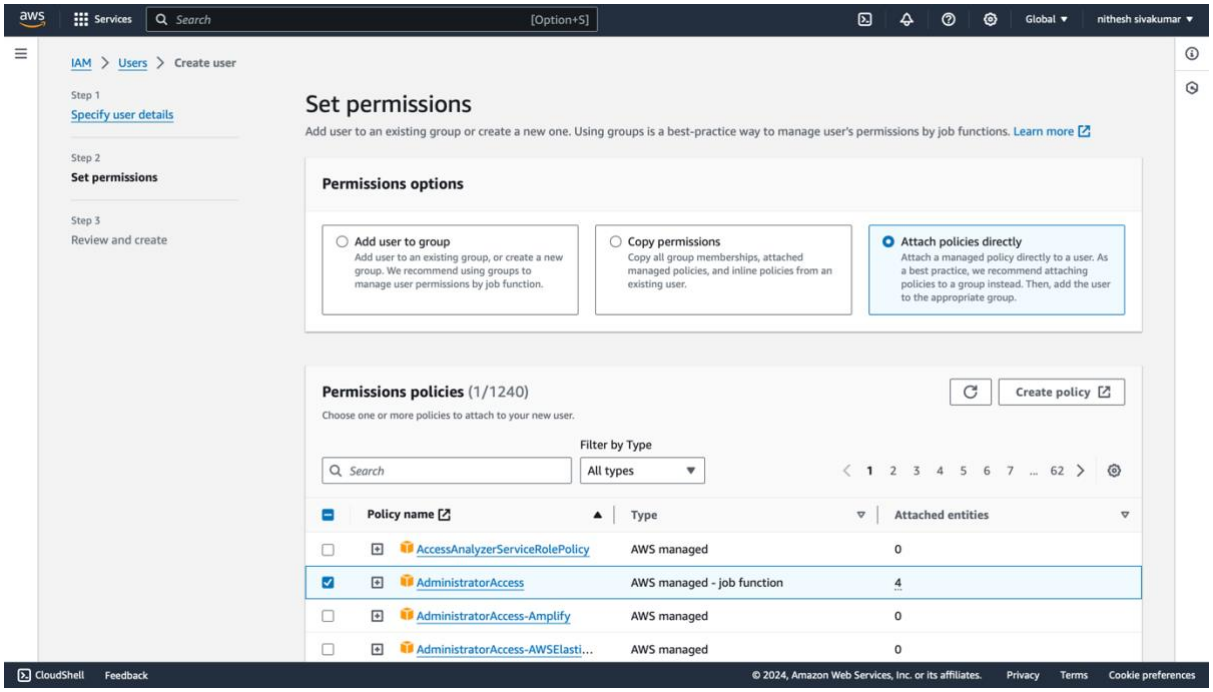
Step-by-Step Implementation Guide

You cannot create **AWS Bedrock** resources if you are logged in as the **root user**. To proceed, sign in with an **IAM user** that has the necessary permissions.

Step 1: Create an IAM User

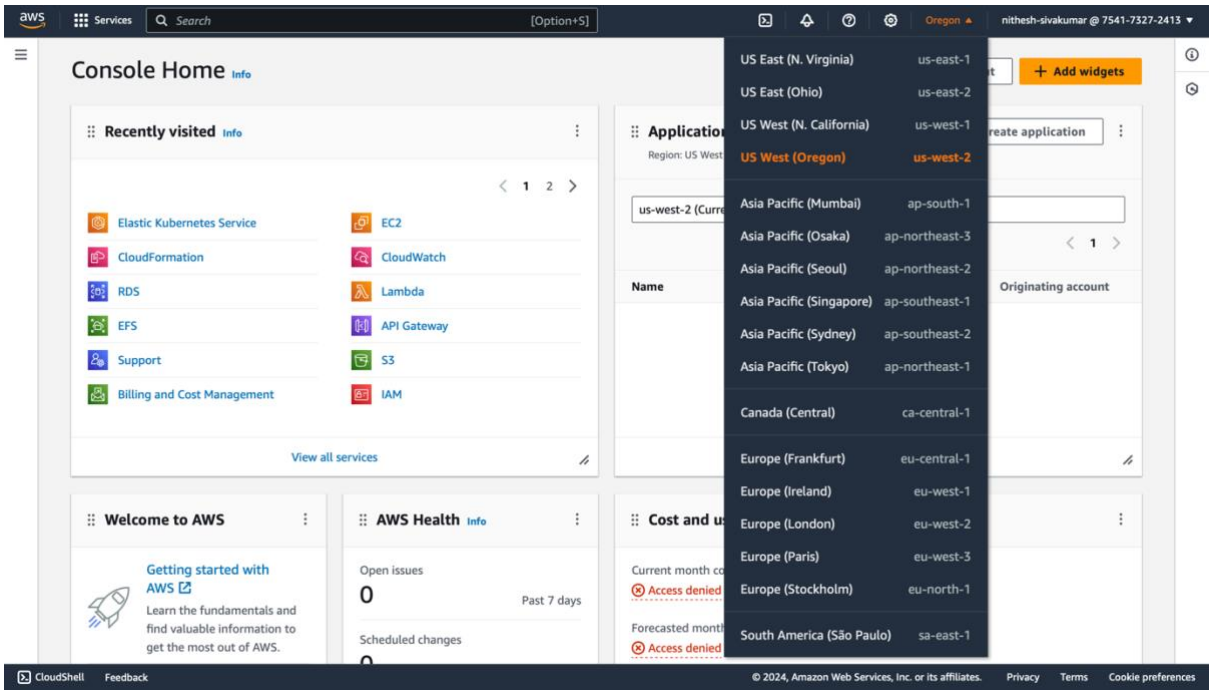
1. Navigate to the AWS Management Console and go to the IAM (Identity and Access Management) service.
2. Click on “Users” in the left sidebar.
3. Select “Create User” and enter a username of your choice.
4. Configure the required permissions:
    - For admin access, attach the AdministratorAccess policy.
5. Enable AWS Management Console access and set a password for the user.
6. Click “Create User” to finalize the setup.





Step 2: Log in as an IAM User

1. Open the AWS Management Console login page.
2. Instead of using the root credentials, select Sign in as an IAM user.
3. Enter your AWS Account ID or alias and click Next.
4. Provide your IAM username and password created in the previous step.
5. Click “Sign In” to access the AWS console with IAM privileges.
6. Select us-west-2 region.

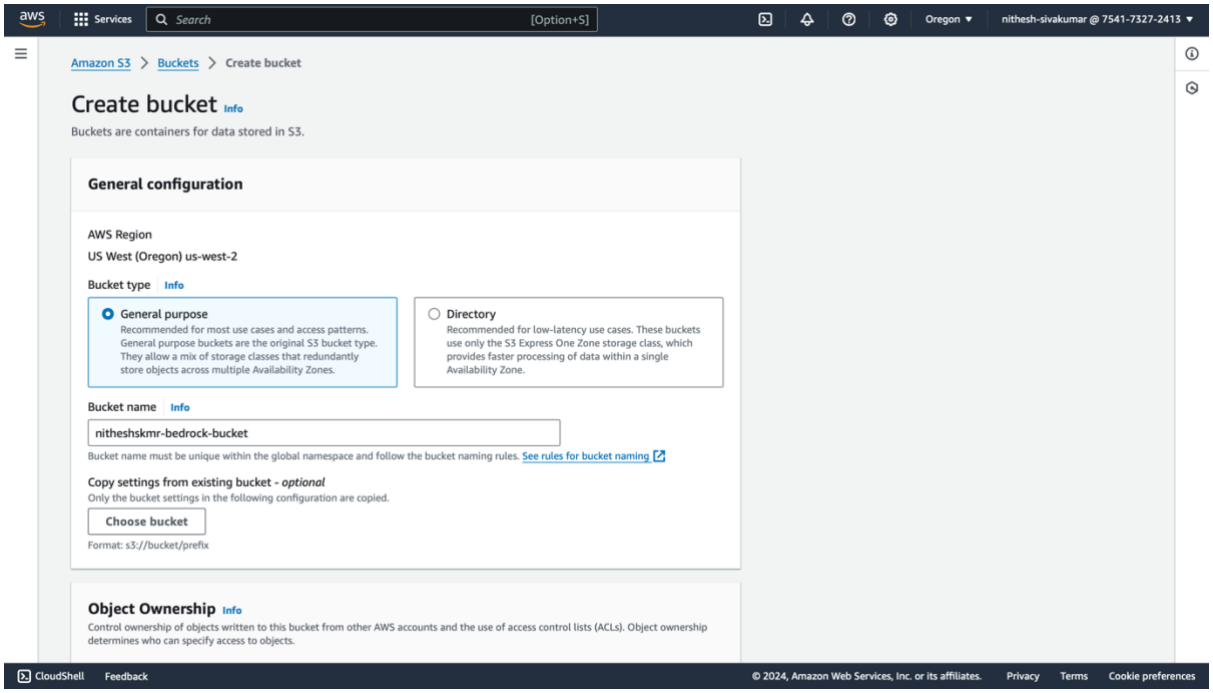


Creating an S3 Bucket

To store the PDFs that will serve as the knowledge base for the application, follow these steps:

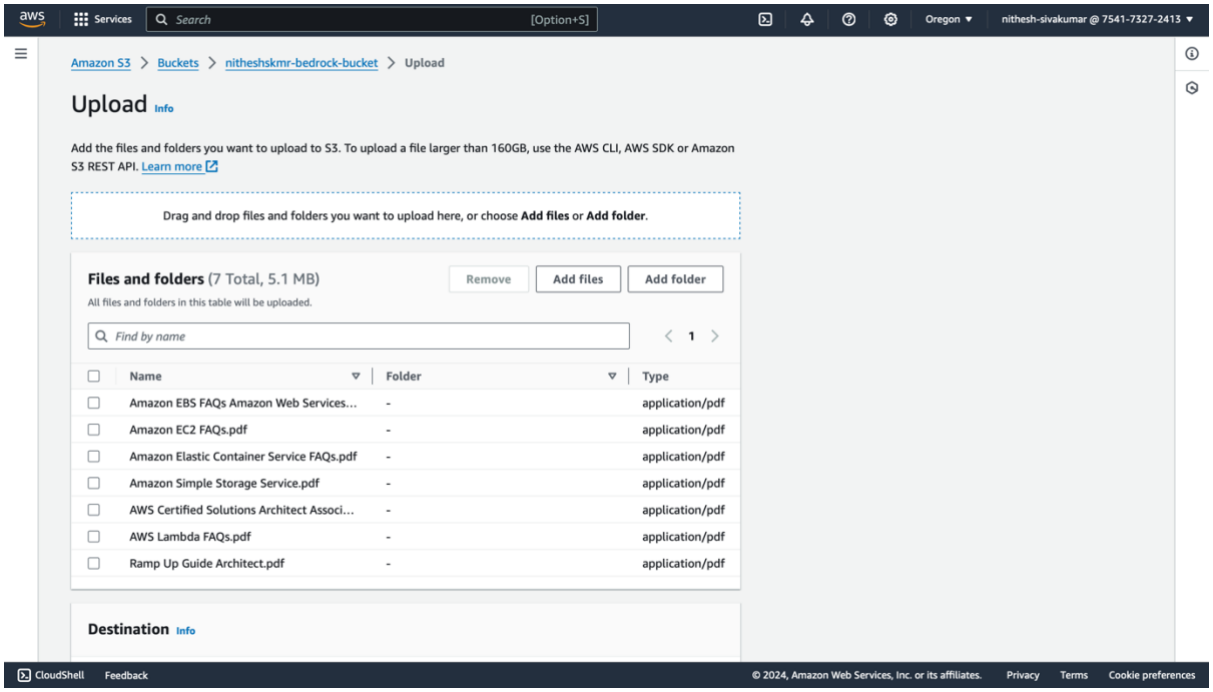
1. In the AWS Management Console, navigate to the S3 service.
2. Click on "Create bucket."
3. Enter a bucket name. The name must be globally unique across all AWS accounts.
4. Choose the AWS region where you want to create the bucket.
5. Configure additional settings such as object ownership and public access settings based on your requirements.
6. Click on "Create bucket" to complete the process.

This bucket will serve as the **source storage** for all PDFs used in the knowledge base. Once created, you can upload the required documents to this bucket.



Uploading PDFs to the S3 Bucket

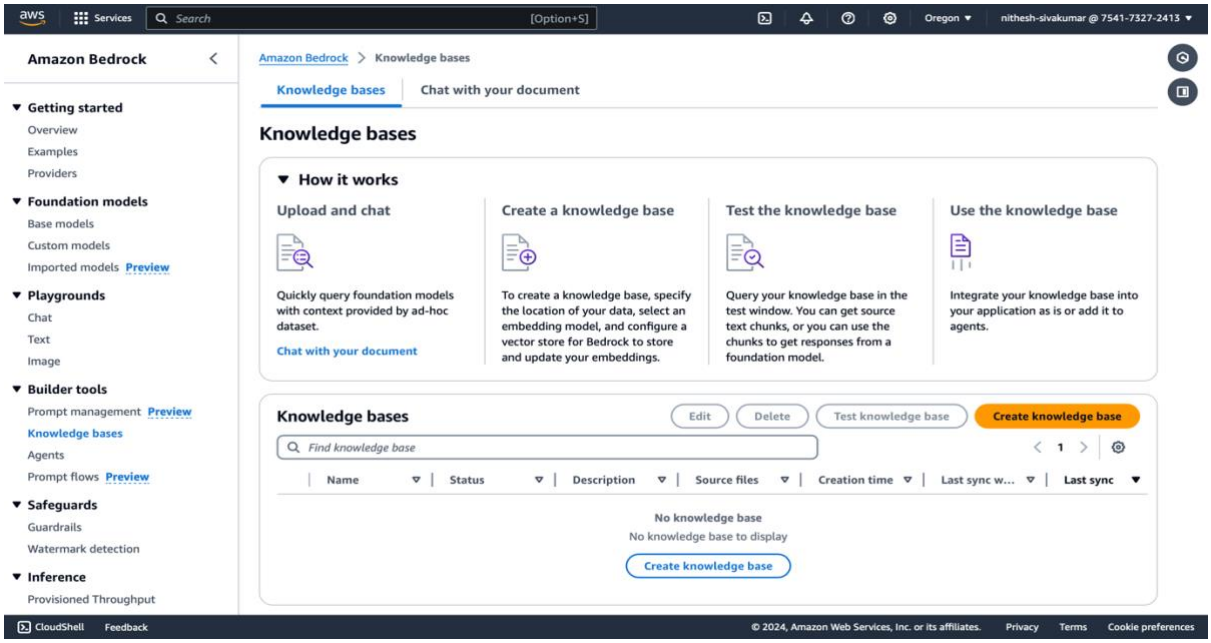
To populate the knowledge base, upload the relevant PDFs and documents to the S3 bucket. Since professor materials are not available, general PDFs have been used for this implementation



Creating a Knowledge Base in AWS Bedrock

To enable the retrieval of information from the uploaded PDFs, a knowledge base needs to be created in AWS Bedrock. Follow these steps:

1. In the AWS Management Console, navigate to **Amazon Bedrock**.
2. Click on **"Knowledge Bases"** from the left menu.
3. Select **"Create Knowledge Base"** to start the setup.



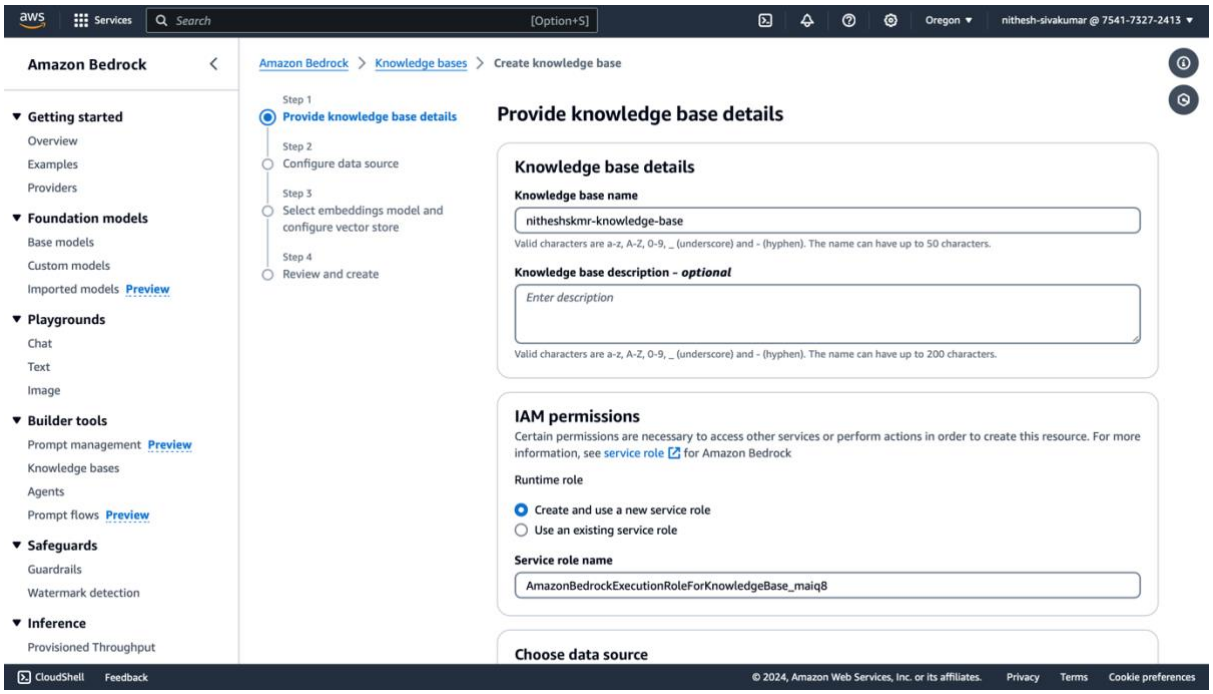
4. Enter a name for the knowledge base.

Configuring IAM Permissions for the Knowledge Base

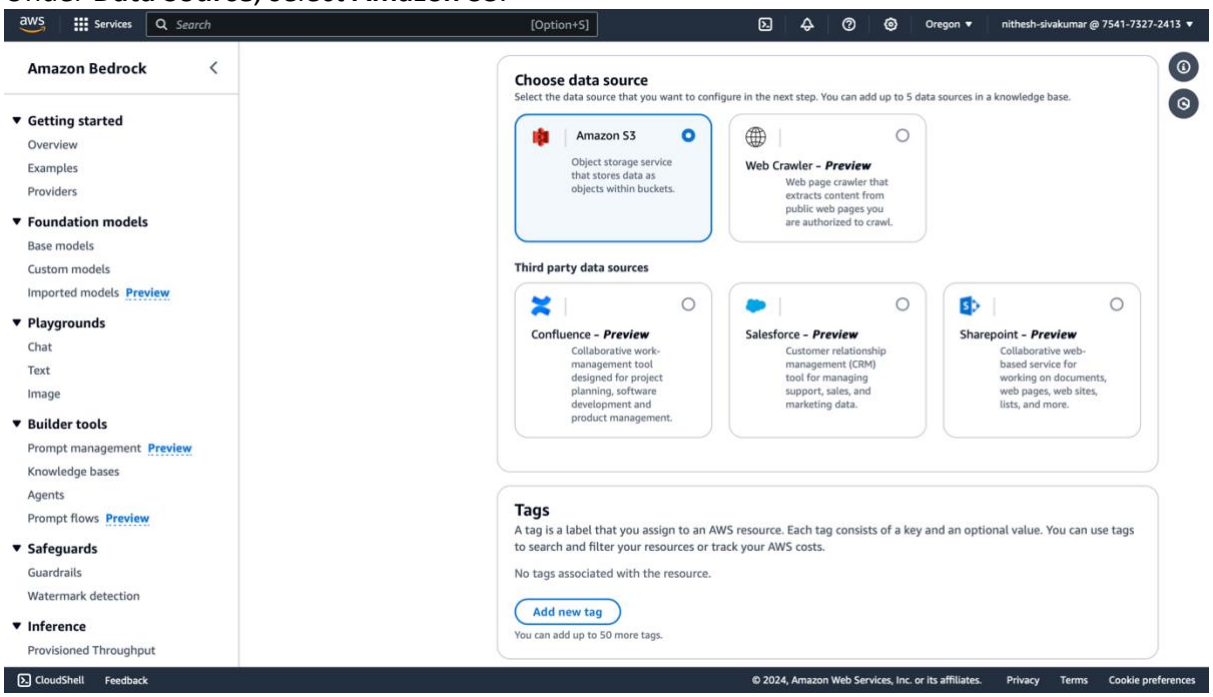
When setting up the Knowledge Base in Amazon Bedrock, you need to grant it access to your S3 bucket and other AWS services.

1. In the **IAM Permissions** section, select **"Create new service role"**.
2. This role allows Bedrock to read documents from S3, process them, and interact with other AWS services securely.
3. Choosing the auto-generated role simplifies setup and ensures the correct permissions are applied.

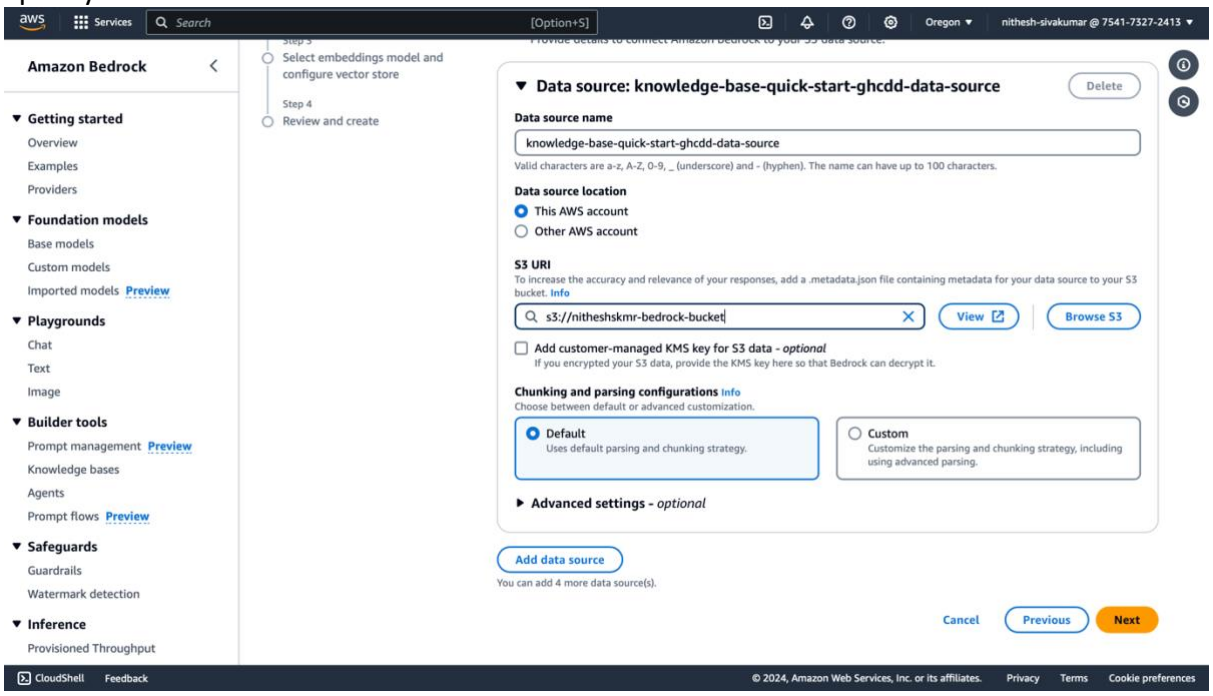
This ensures Bedrock can retrieve and process knowledge base data securely.



Under **Data Source**, select **Amazon S3**.

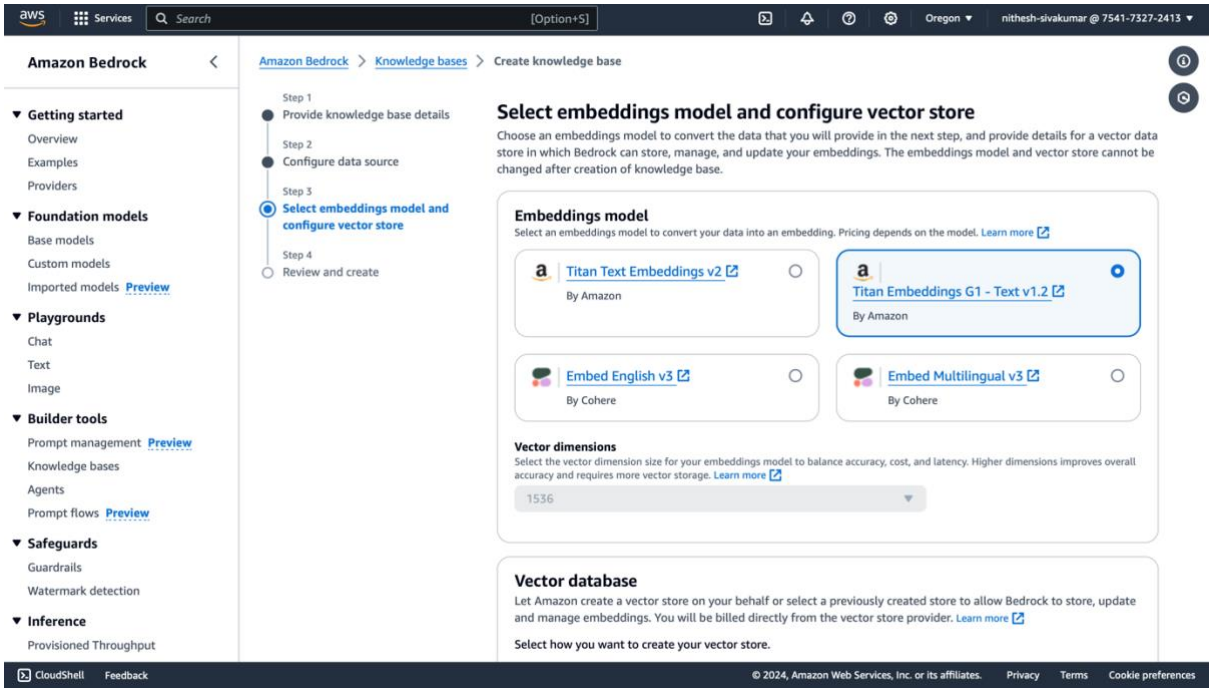


Specify the S3 bucket where the PDFs are stored.

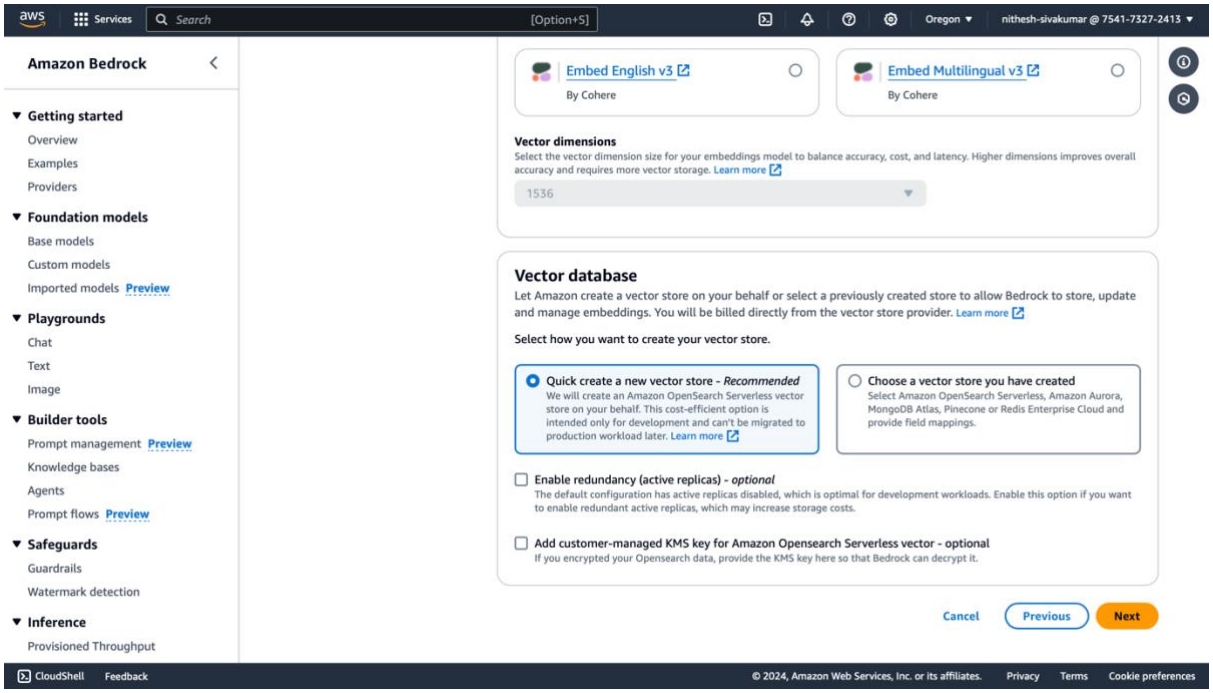


Click next, then select **Titan Embeddings** as the embedding model.

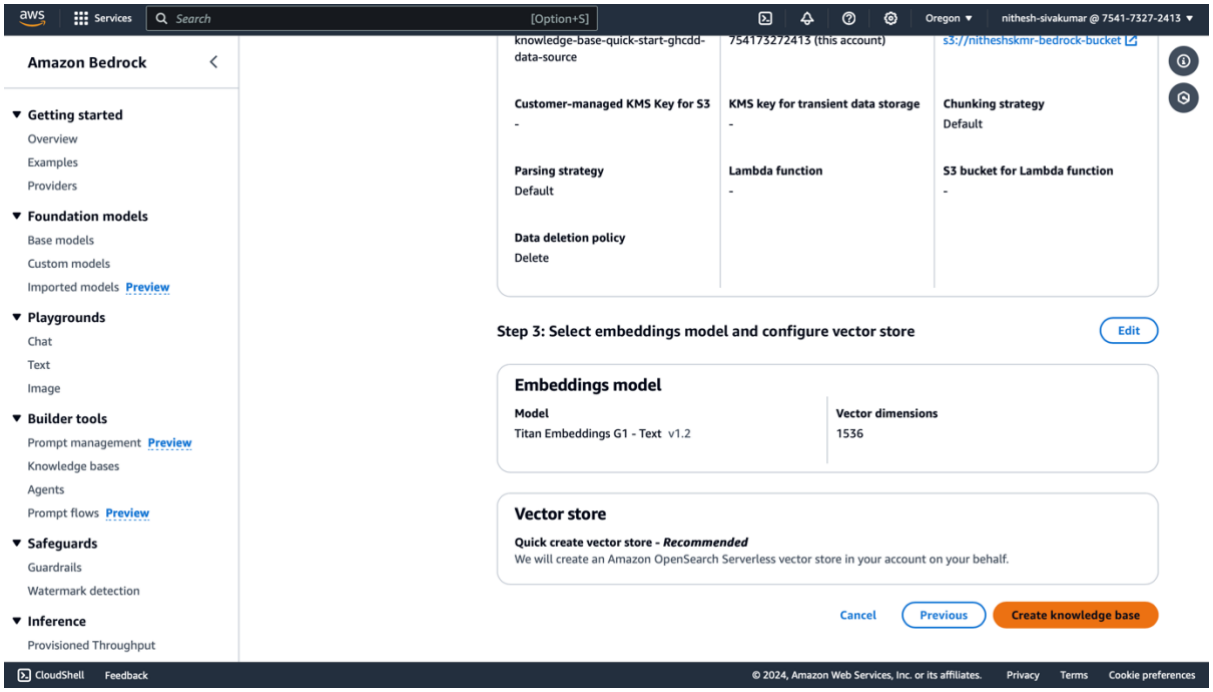
An embedding model converts text into numerical representations, allowing efficient searching and retrieval of relevant information. It helps the knowledge base understand and match user queries with the most relevant content from the uploaded documents.



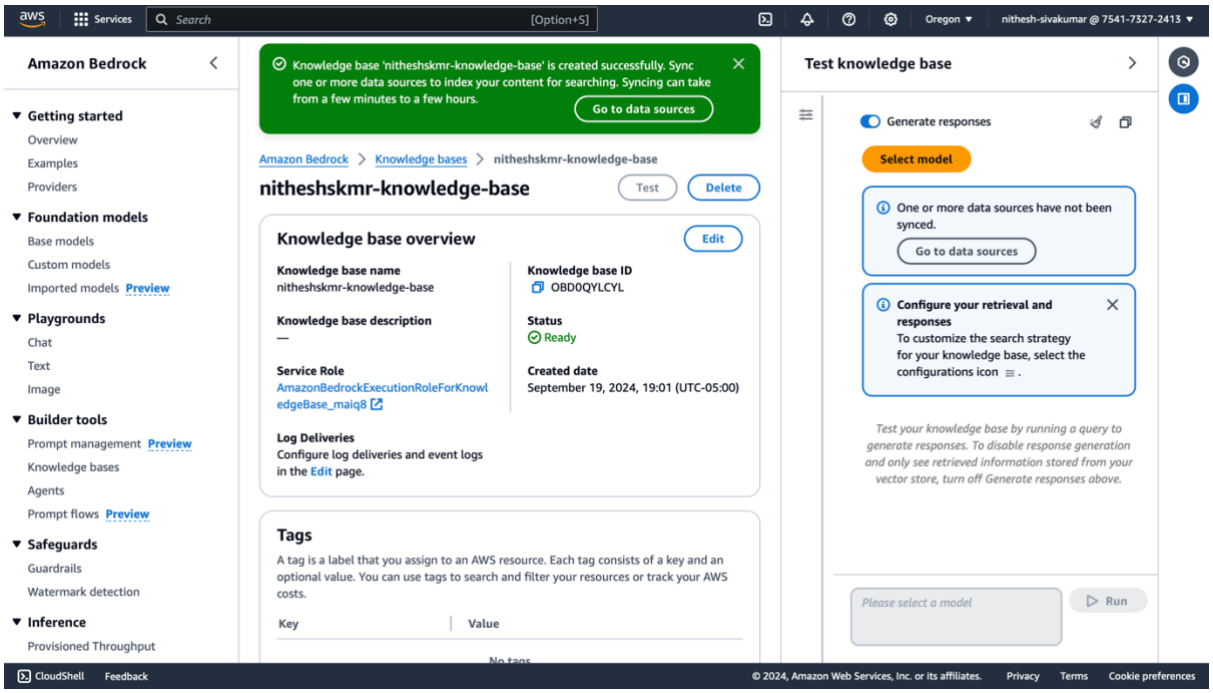
For the vector database, click **Create new vector database** to store and index the processed document embeddings for efficient retrieval. An AWS Opensearch DB will be created.



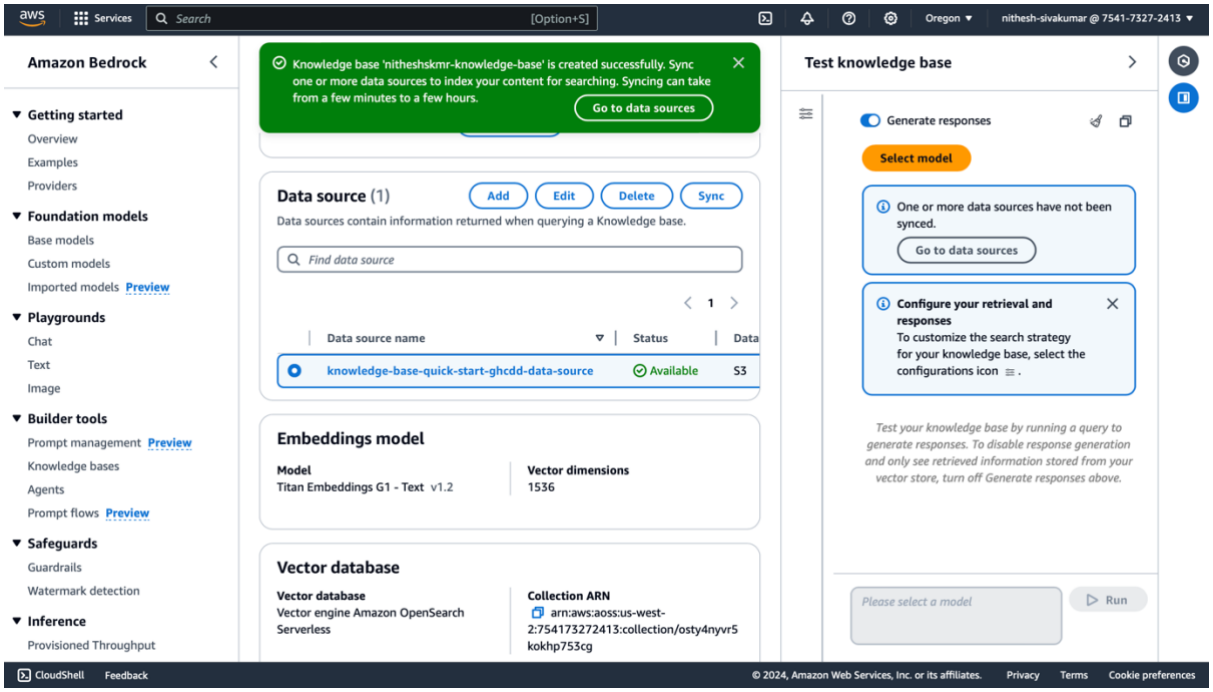
Click next, review all the settings, and then click **Create Knowledge Base**. The creation process will take approximately 4-5 minutes to complete.

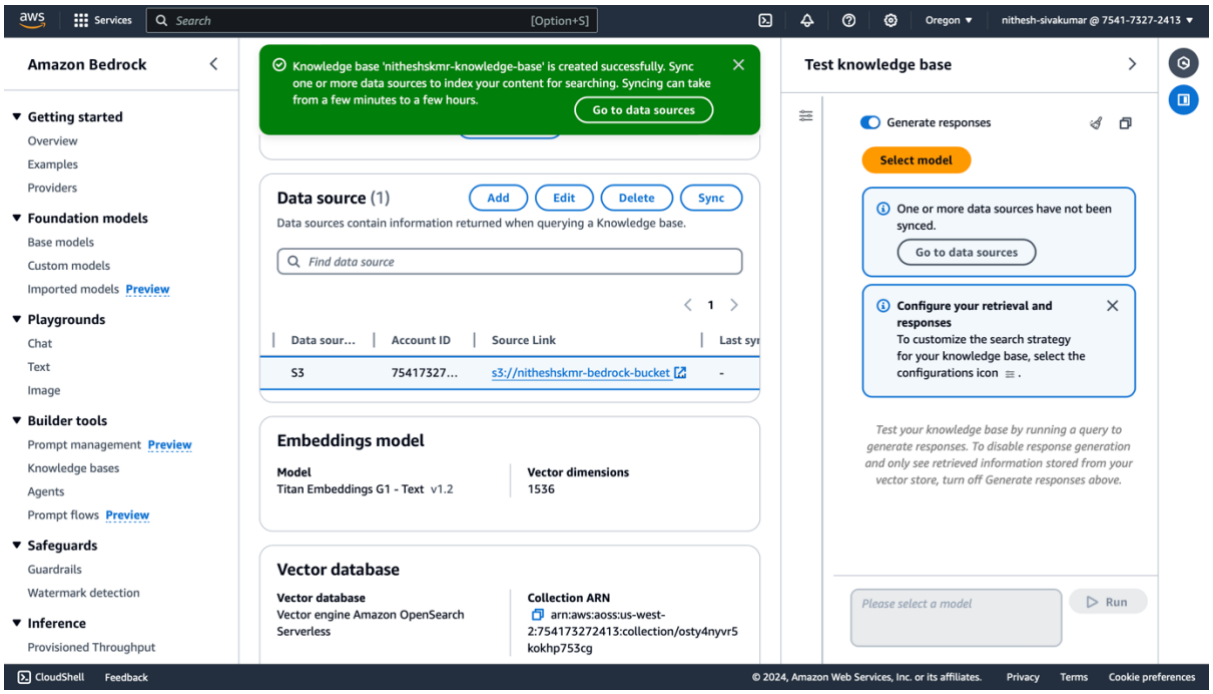


You need to sync your data to make it available for retrieval. Click on **Go to Data Source**, then select **Sync** to process the uploaded documents.



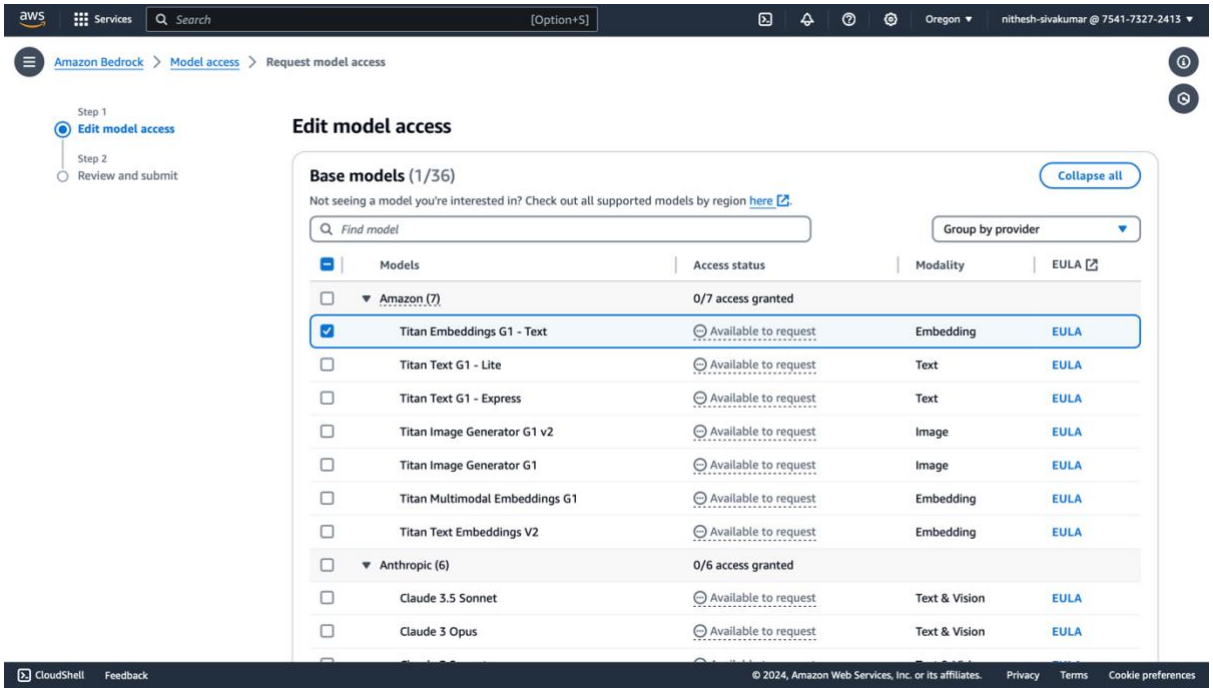
You can now see the **data source name** listed under the **Data Source** section, which is linked to the S3 bucket where the documents are stored.



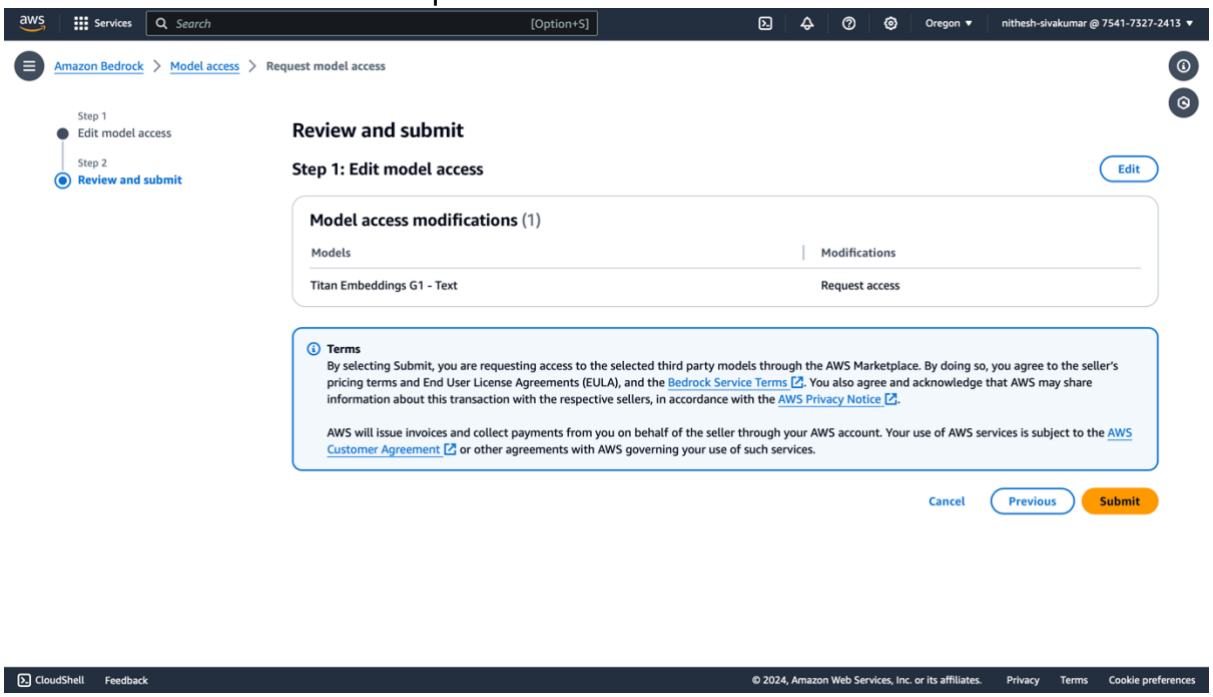


Meanwhile, you need access to **Amazon Titan Embeddings G1 - Text**. If you don't have access, request it through the **Service Quotas** or **Bedrock Access Request** page in the AWS Management Console.

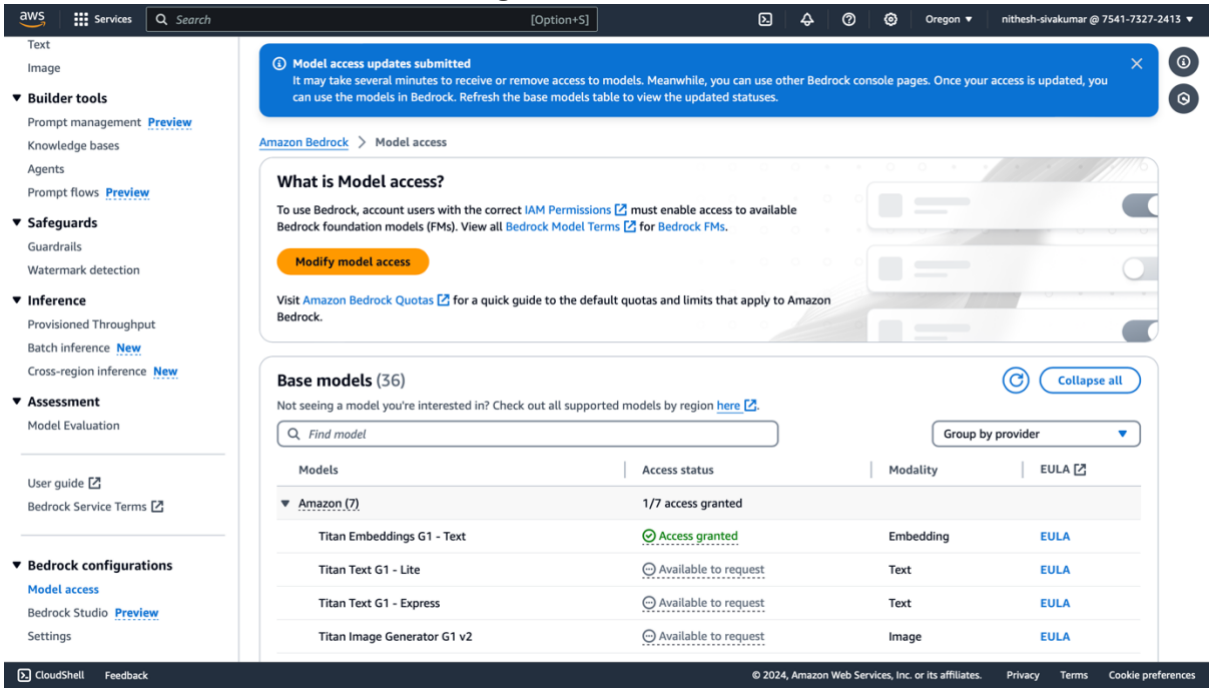
Amazon Titan Embeddings G1 - Text is used for efficient text retrieval by converting documents into searchable vectors. It ensures accurate, scalable, and AWS-optimized search results.



Click **Submit** to finalize the request.

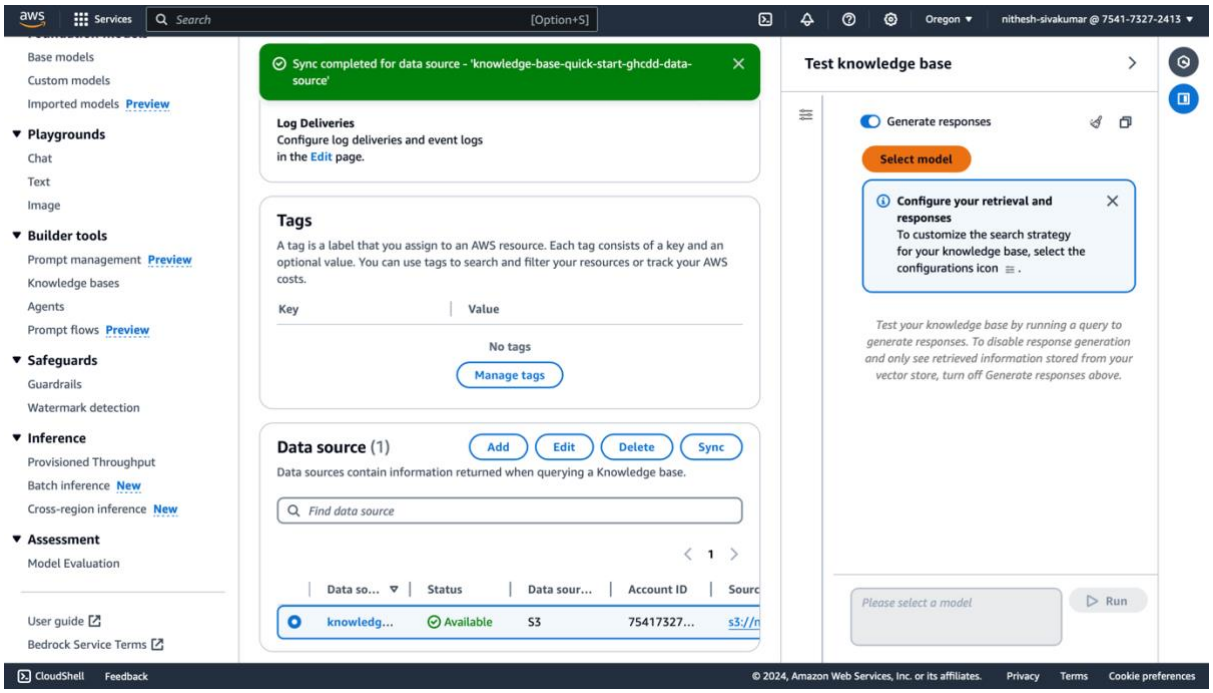


You can see that access has been granted.



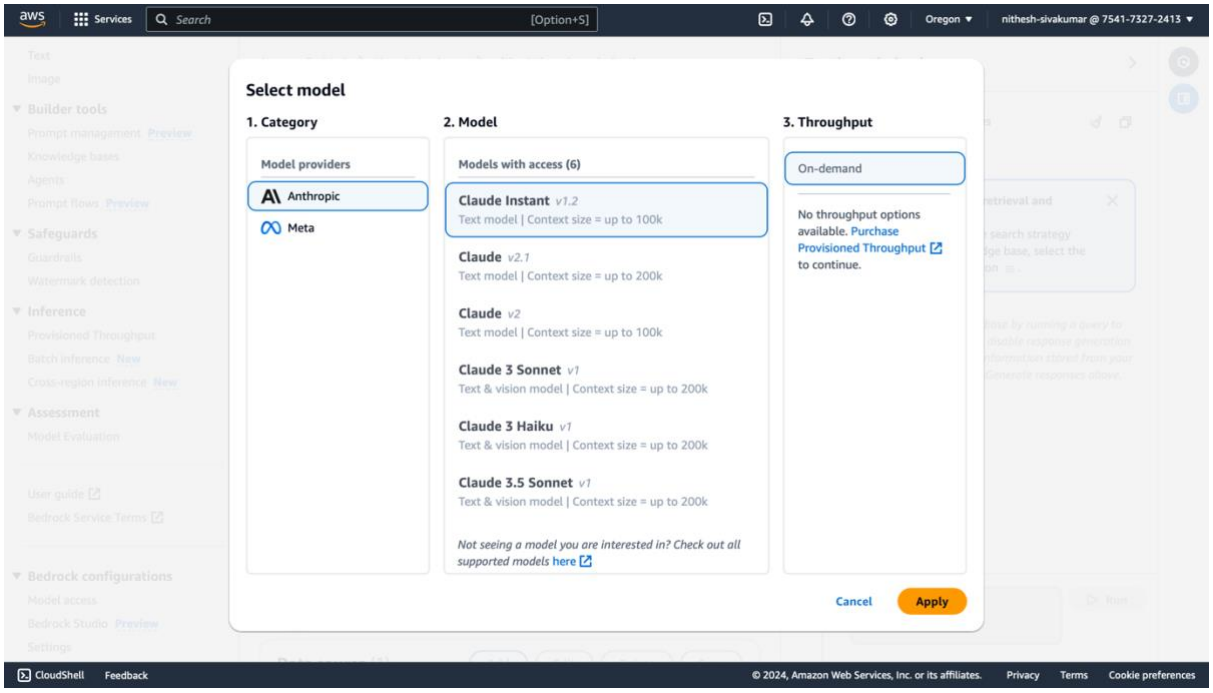
Next, you need to select the foundation model. Click **Select Model** to proceed.

A foundation model processes user queries and generates responses based on retrieved knowledge. It enhances accuracy by leveraging AI to understand and summarize relevant information.

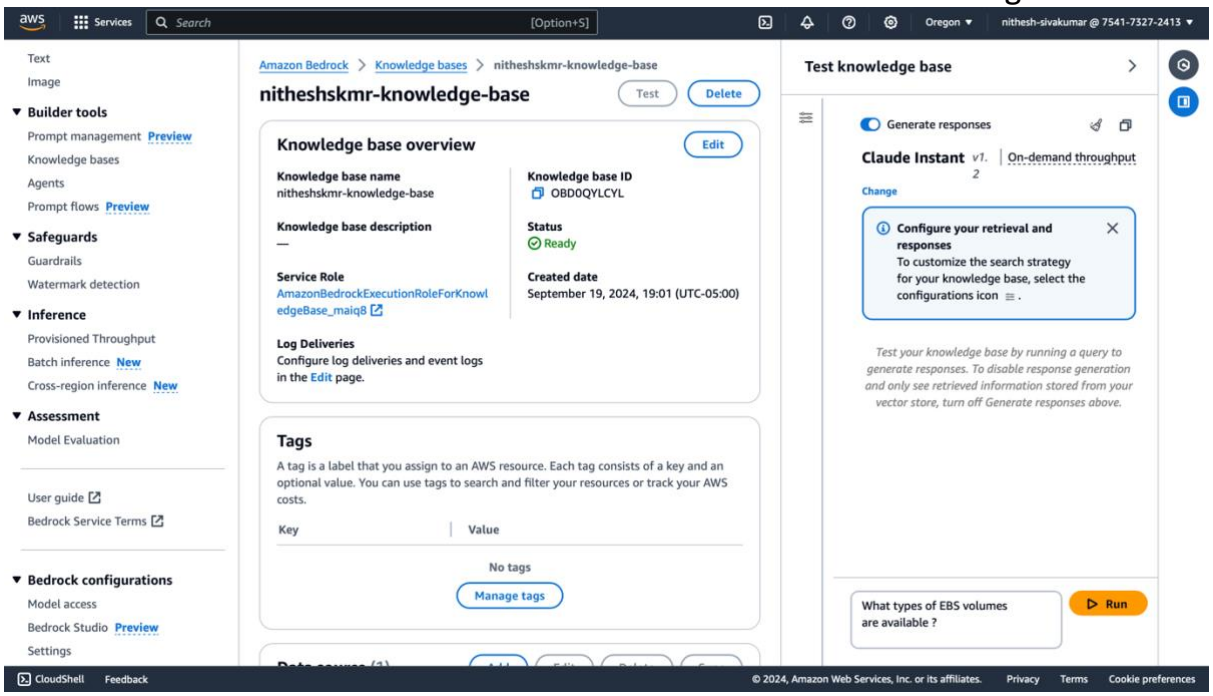


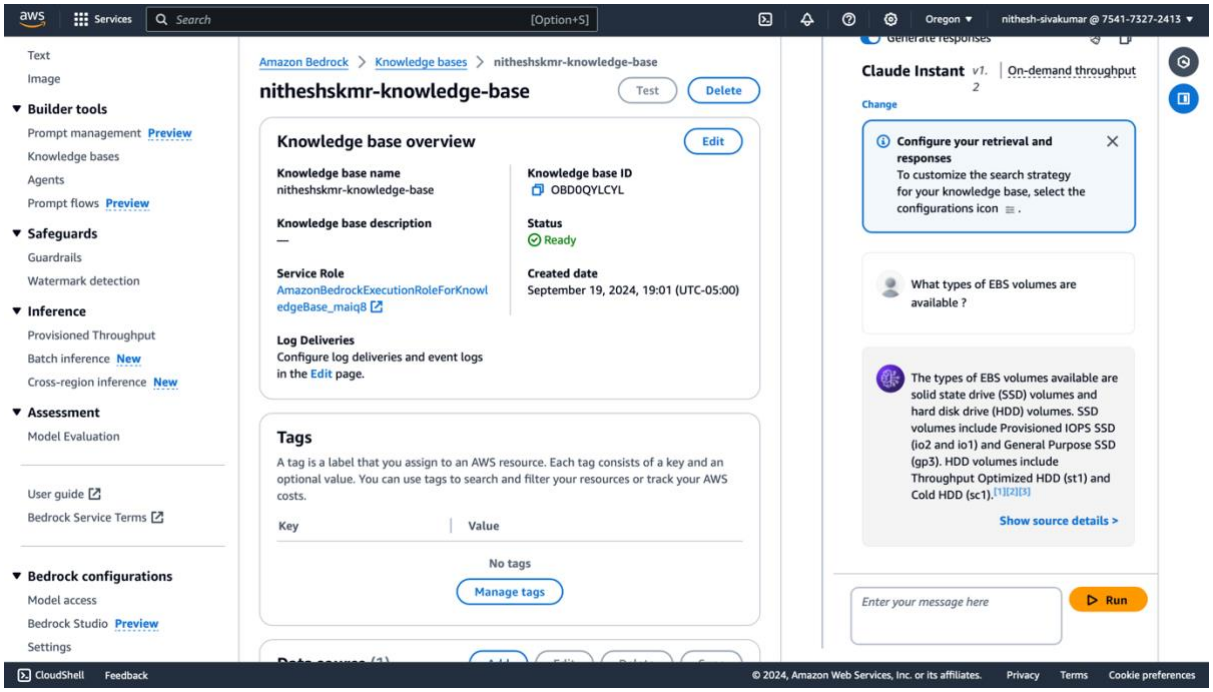
Currently, only **Anthropic and Meta models** are available for access via the GUI. For testing purposes, we can use one of these models to check if it retrieves information from our S3 bucket.

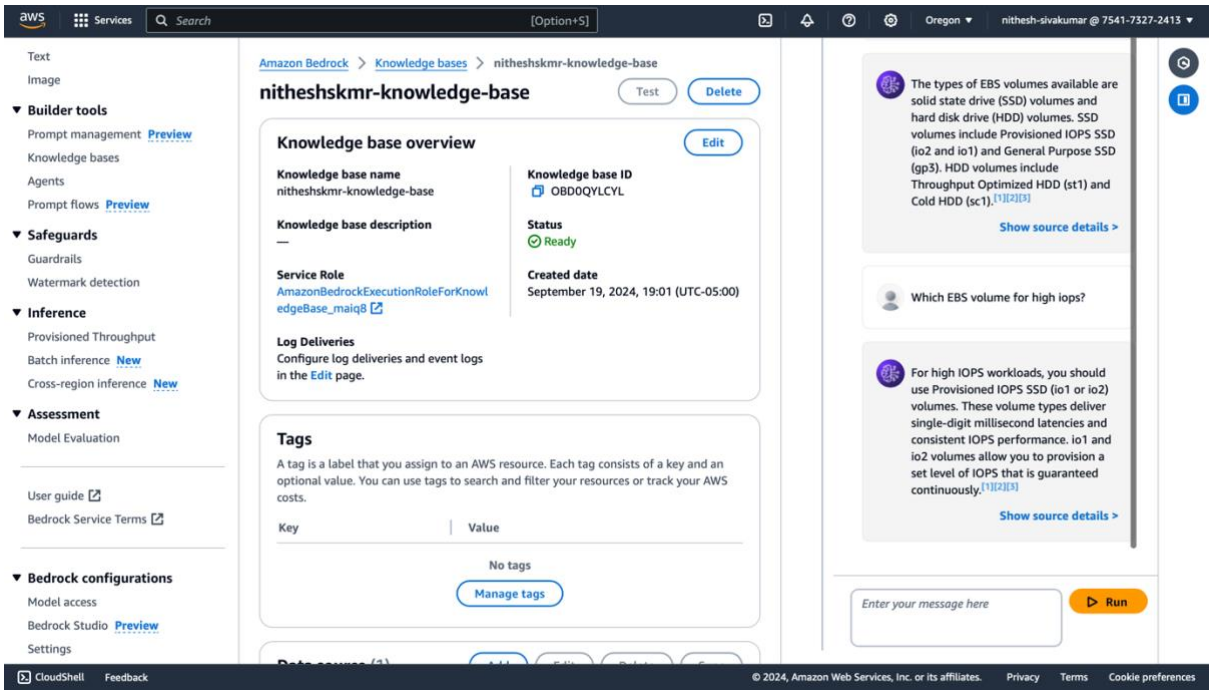
I will select **Claude**, then click **Apply**.



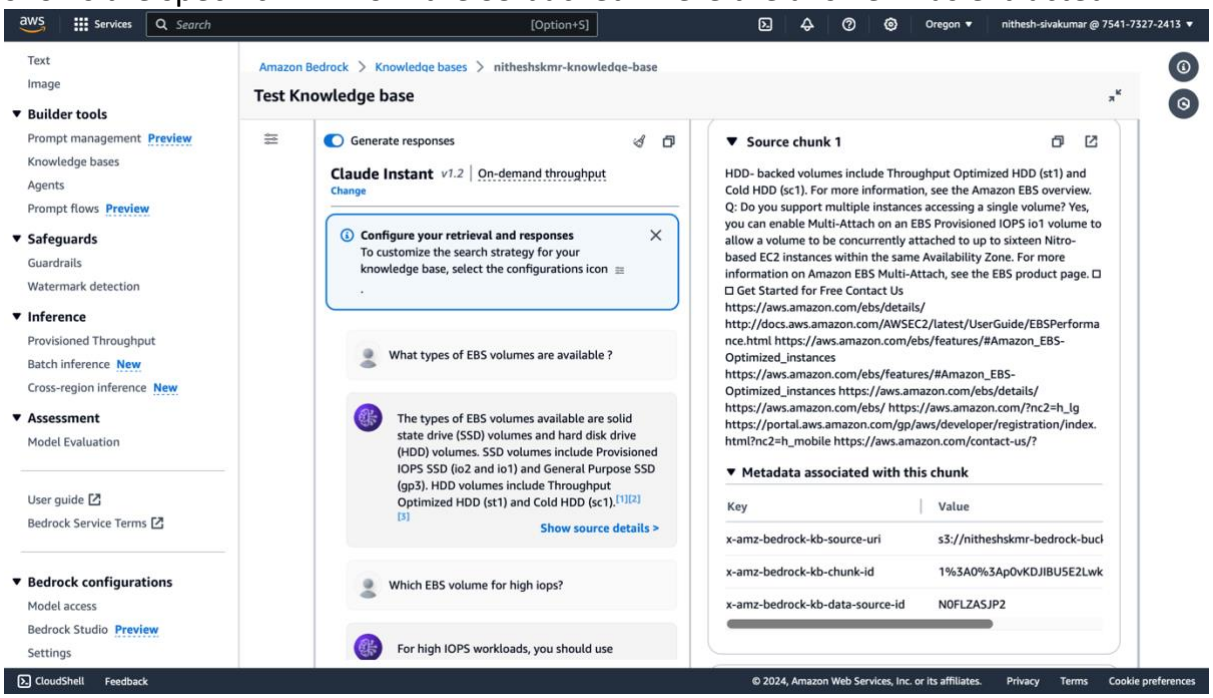
I am asking a question related to **EBS**, which was included in the uploaded PDFs, to test if the model retrieves the correct information from the knowledge base.




aws
Services
Search
[Option+S]
Oregon
nithesh-sivakumar @ 7541-7327-2413
Text
Image
Builder tools
Prompt management Preview
Knowledge bases
Agents
Prompt flows Preview
Safeguards
Guardrails
Watermark detection
Inference
Provisioned Throughput
Batch inference New
Cross-region inference New
Assessment
Model Evaluation
User guide
Bedrock Service Terms
Bedrock configurations
Model access
Bedrock Studio Preview
Settings
Amazon Bedrock > Knowledge bases > nitheshskmr-knowledge-base
nitheshskmr-knowledge-base
Test
Delete
Knowledge base overview
Edit
Knowledge base name
nitheshskmr-knowledge-base
Knowledge base ID
OBD0QYLCYL
Knowledge base description
—
Status
Ready
Service Role
AmazonBedrockExecutionRoleForKnowledgeBase_maiq8
Created date
September 19, 2024, 19:01 (UTC-05:00)
Log Deliveries
Configure log deliveries and event logs in the Edit page.
Tags
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
No tags
Manage tags
Claude Instant v1.2
On-demand throughput
Change
Configure your retrieval and responses
To customize the search strategy for your knowledge base, select the configurations icon .
What types of EBS volumes are available ?
The types of EBS volumes available are solid state drive (SSD) volumes and hard disk drive (HDD) volumes. SSD volumes include Provisioned IOPS SSD (io2 and io1) and General Purpose SSD (gp3). HDD volumes include Throughput Optimized HDD (st1) and Cold HDD (sc1).[1][2][3]
Show source details >
Enter your message here
Run
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Option+S]
Oregon
nithesh-sivakumar @ 7541-7327-2413
Text
Image
Builder tools
Prompt management Preview
Knowledge bases
Agents
Prompt flows Preview
Safeguards
Guardrails
Watermark detection
Inference
Provisioned Throughput
Batch inference New
Cross-region inference New
Assessment
Model Evaluation
User guide
Bedrock Service Terms
Bedrock configurations
Model access
Bedrock Studio Preview
Settings
Amazon Bedrock > Knowledge bases > nitheshskmr-knowledge-base
nitheshskmr-knowledge-base
Test
Delete
Knowledge base overview
Edit
Knowledge base name
nitheshskmr-knowledge-base
Knowledge base ID
OBD0QYLCYL
Knowledge base description
—
Status
Ready
Service Role
AmazonBedrockExecutionRoleForKnowledgeBase_maiq8
Created date
September 19, 2024, 19:01 (UTC-05:00)
Log Deliveries
Configure log deliveries and event logs in the Edit page.
Tags
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
No tags
Manage tags
The types of EBS volumes available are solid state drive (SSD) volumes and hard disk drive (HDD) volumes. SSD volumes include Provisioned IOPS SSD (io2 and io1) and General Purpose SSD (gp3). HDD volumes include Throughput Optimized HDD (st1) and Cold HDD (sc1).[1][2][3]
Show source details >
Which EBS volume for high iops?
For high IOPS workloads, you should use Provisioned IOPS SSD (io1 or io2) volumes. These volume types deliver single-digit millisecond latencies and consistent IOPS performance. io1 and io2 volumes allow you to provision a set level of IOPS that is guaranteed continuously.[1][2][3]
Show source details >
Enter your message here
Run
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

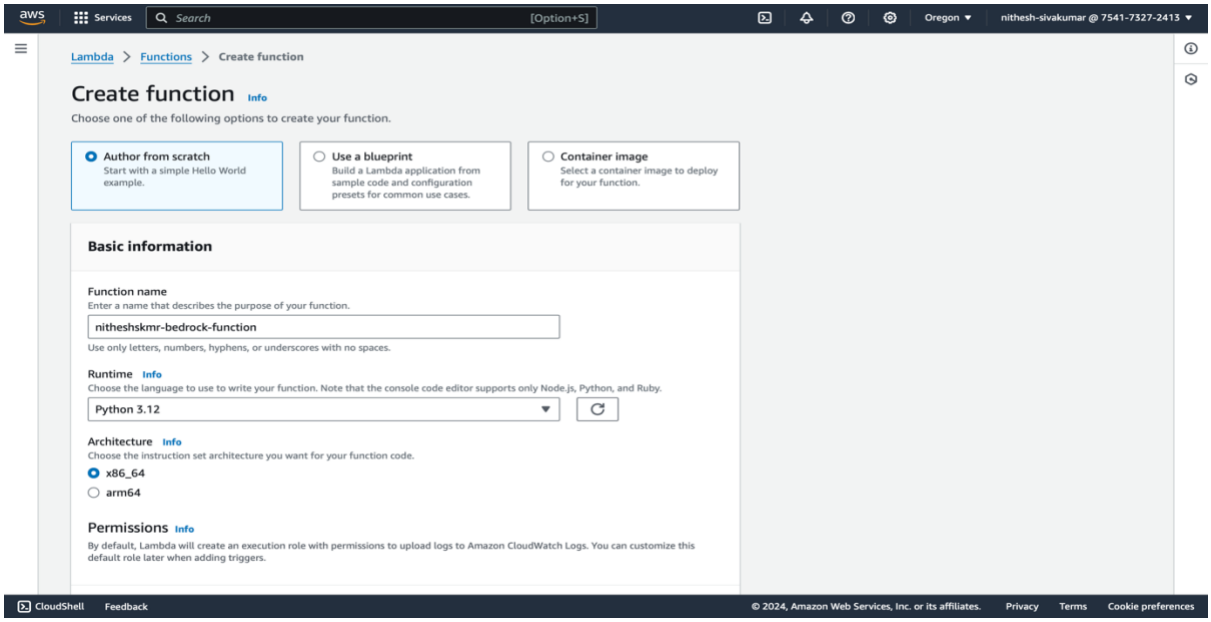
You can also verify the source of the retrieved information. Under **Source Details**, it shows the specific PDF from the S3 bucket where the answer was extracted.



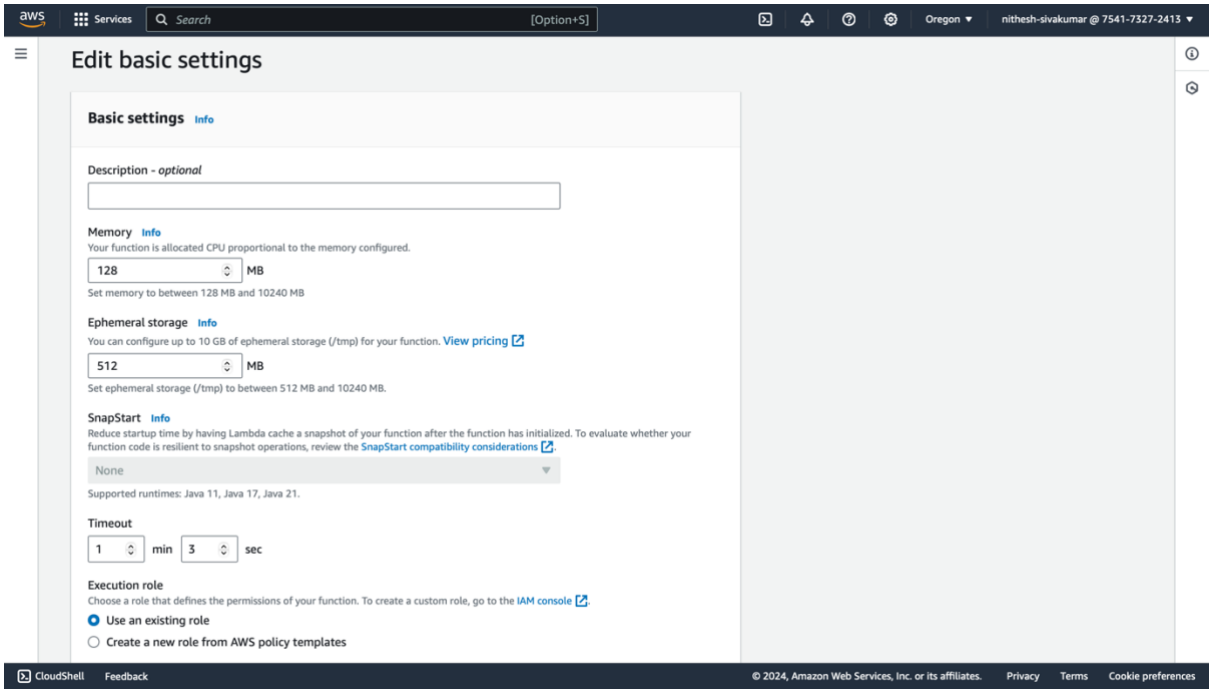
Setting Up the Retrieve and Generate API

Next, you need to create a Lambda function to handle API requests.

1. Go to **AWS Lambda** in the console.
2. Click **Create Function**.
3. Enter a **function name** (e.g., RetrieveGenerateFunction).
4. Select **Runtime** as **Python**.
5. Click **Create** to proceed.

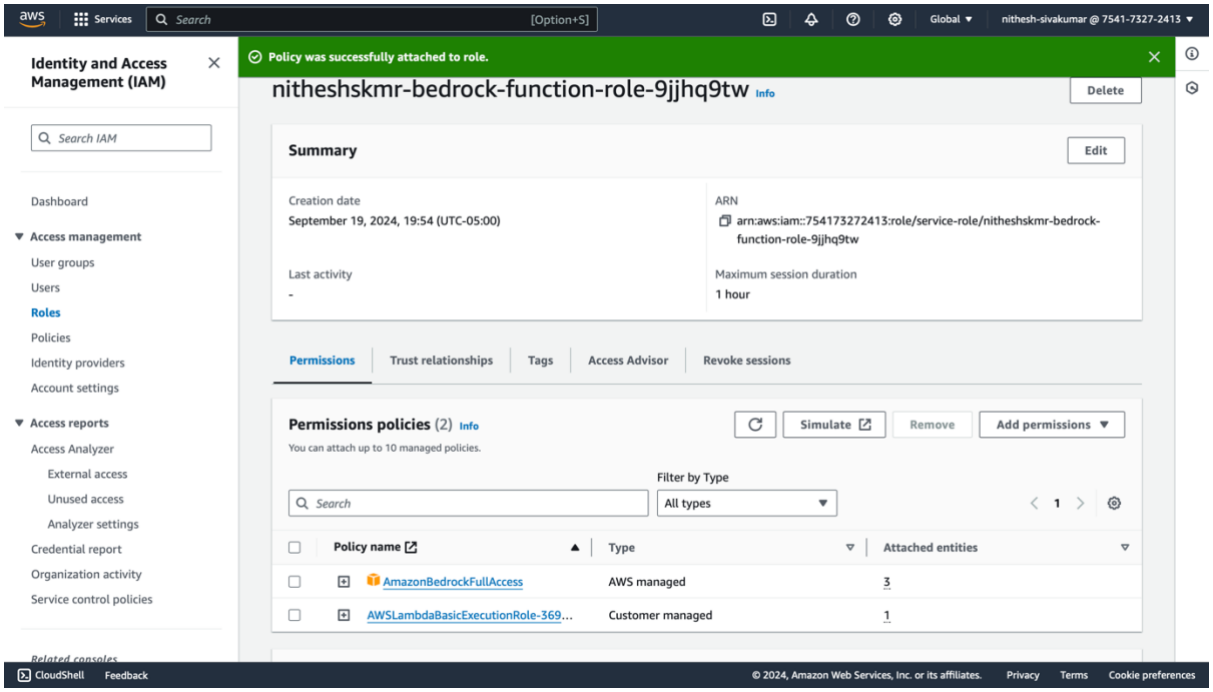


Edit the Lambda function by going to **Configuration**. The default timeout is set to 3 seconds, but for this use case, increase it to 1 minute to prevent timeouts.
Click **Save** to apply the changes.



Next, grant the Lambda function permission to access Bedrock.

1. Go to **Configuration** → **Permissions**.
2. Click on the **Execution Role**, which will open the IAM dashboard.
3. Click on **Add Permissions** and search for **BedrockFullAccess**.
4. Select it and apply the changes.

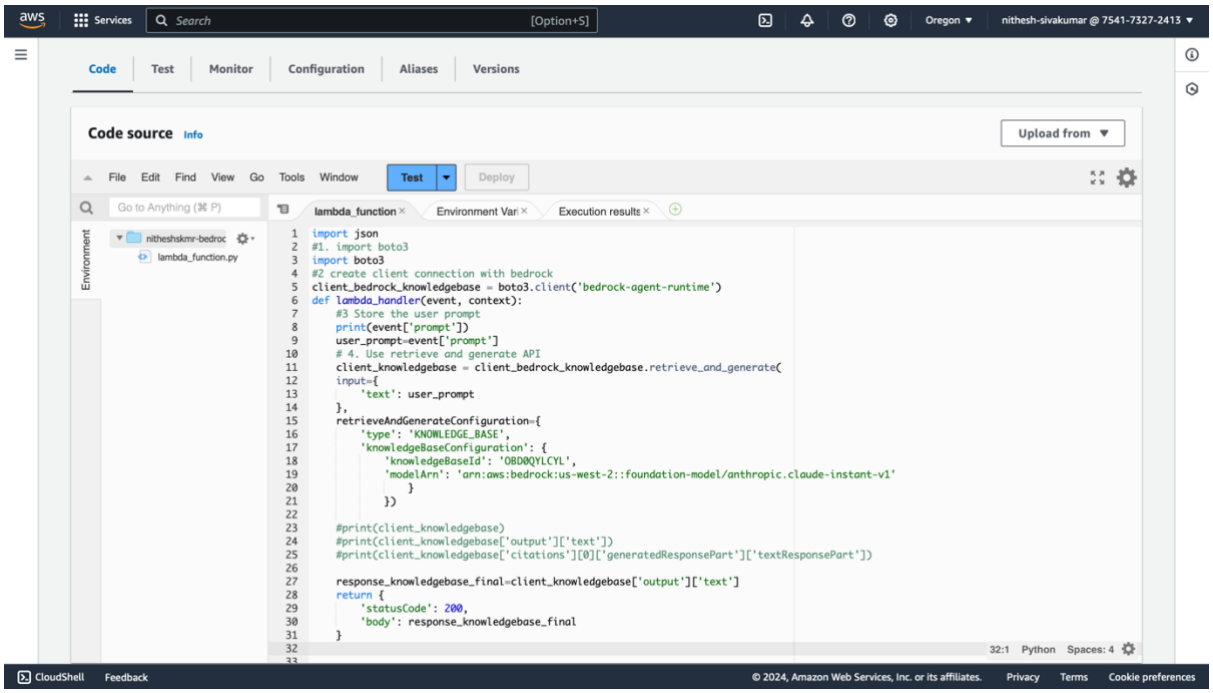


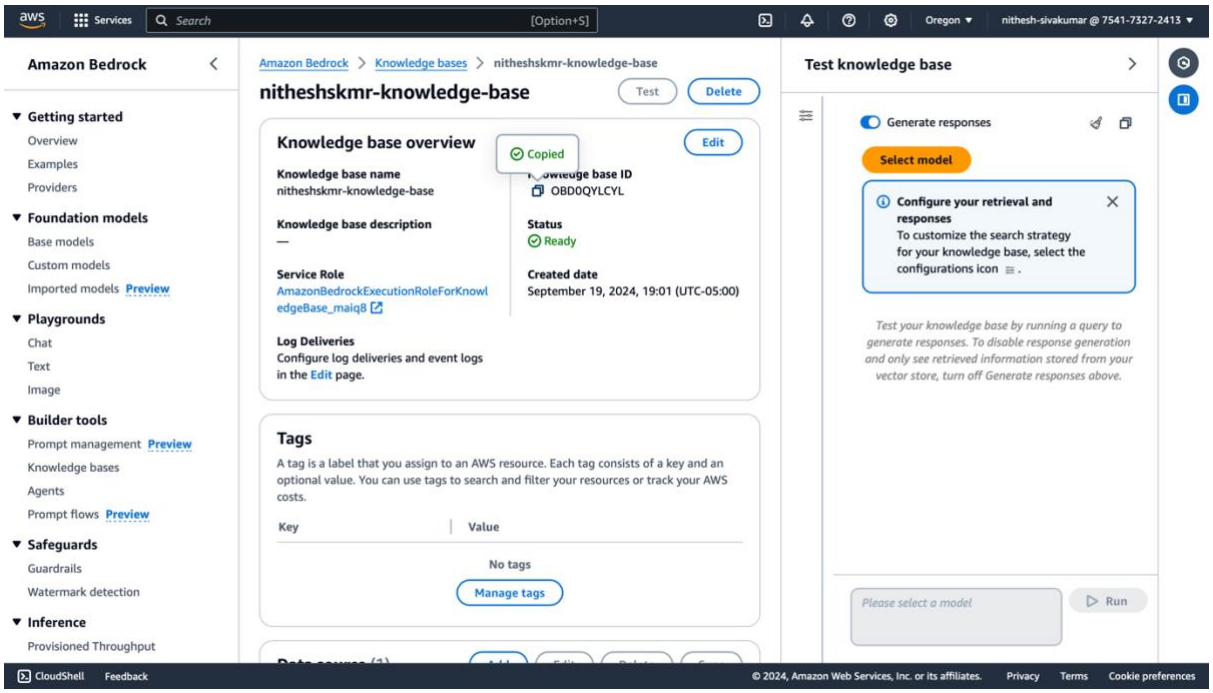
Go back to the **Lambda function**, then under the **Code** section, copy and paste the provided code.

```
import json
#1. import boto3
import boto3
#2 create client connection with bedrock
client_bedrock_knowledgebase = boto3.client('bedrock-agent-runtime')
def lambda_handler(event, context):
    #3 Store the user prompt
    print(event['prompt'])
    user_prompt=event['prompt']
    # 4. Use retrieve and generate API
    client_knowledgebase =
client_bedrock_knowledgebase.retrieve_and_generate(
    input={
        'text': user_prompt
    },
    retrieveAndGenerateConfiguration={
        'type': 'KNOWLEDGE_BASE',
        'knowledgeBaseConfiguration': {
            'knowledgeBaseId': 'O41RCIQ46A',
            'modelArn': 'arn:aws:bedrock:us-west-2::foundation-
model/anthropic.claude-instant-v1'
                }
            })

    # print(client_knowledgebase)
    #print(client_knowledgebase['output']['text'])

#print(client_knowledgebase['citations'][0]['generatedResponsePart']['te
xtResponsePart'])
    response_kbase_final=client_knowledgebase['output']['text']
    return {
        'statusCode': 200,
        'body': response_kbase_final
    }
```

In the code, you need to update the **Knowledge Base ID** to ensure it connects to the correct data source.

1. Go to **Amazon Bedrock**.
2. Navigate to your **Knowledge Base**.
3. Copy the **Knowledge Base ID**.
4. Paste it in the line 18 of your Lambda function code.
5. Deploy the updated code.



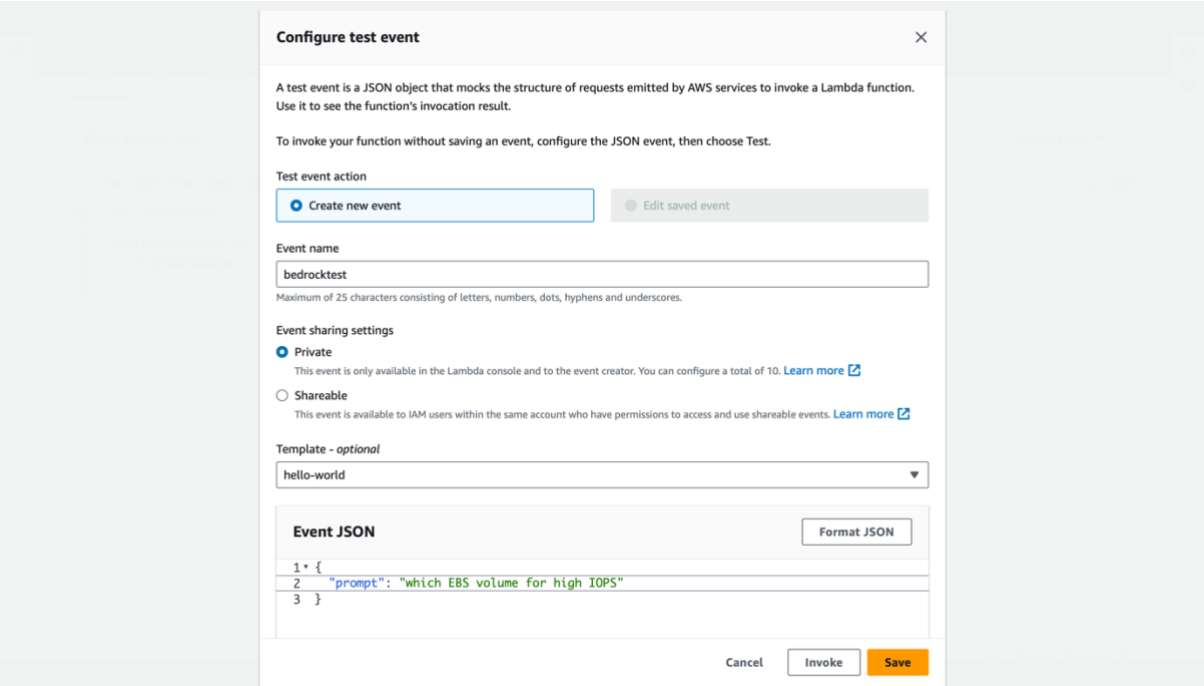


Click **Deploy**, then create and run a **Test Event** to verify the function's execution.

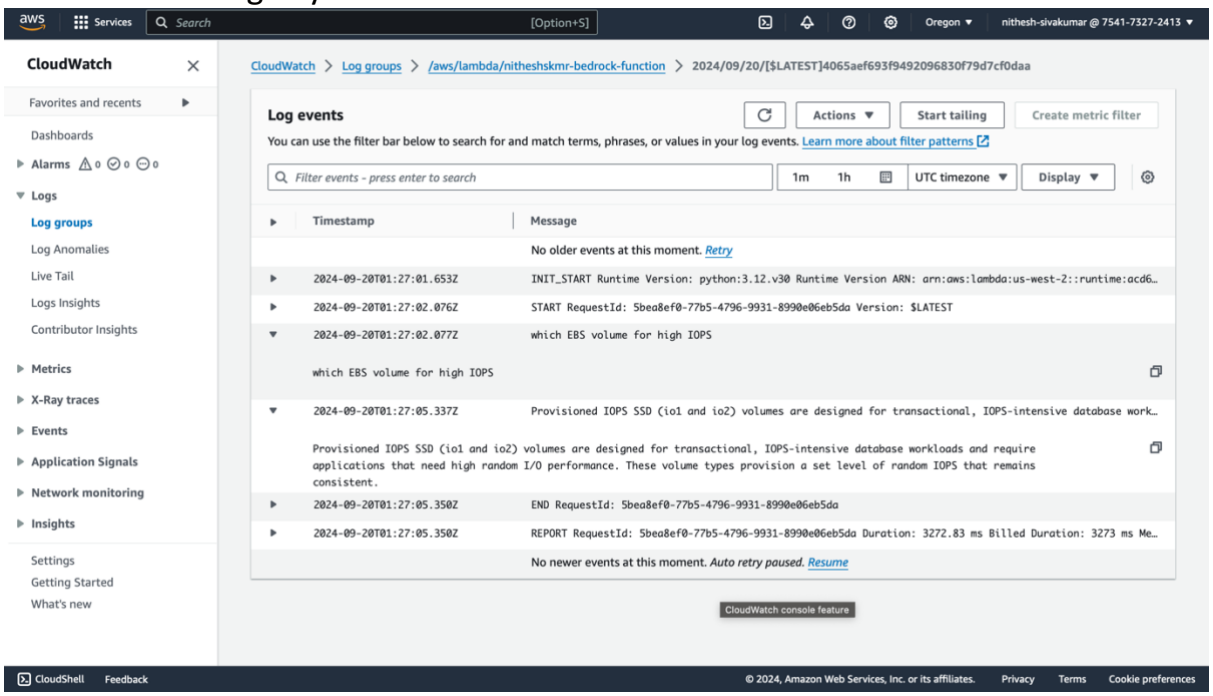
Create a test event with the following details:

- **Test Name:** bedrocktest
- **Prompt:** *Which EBS volume is best for high IOPS?*

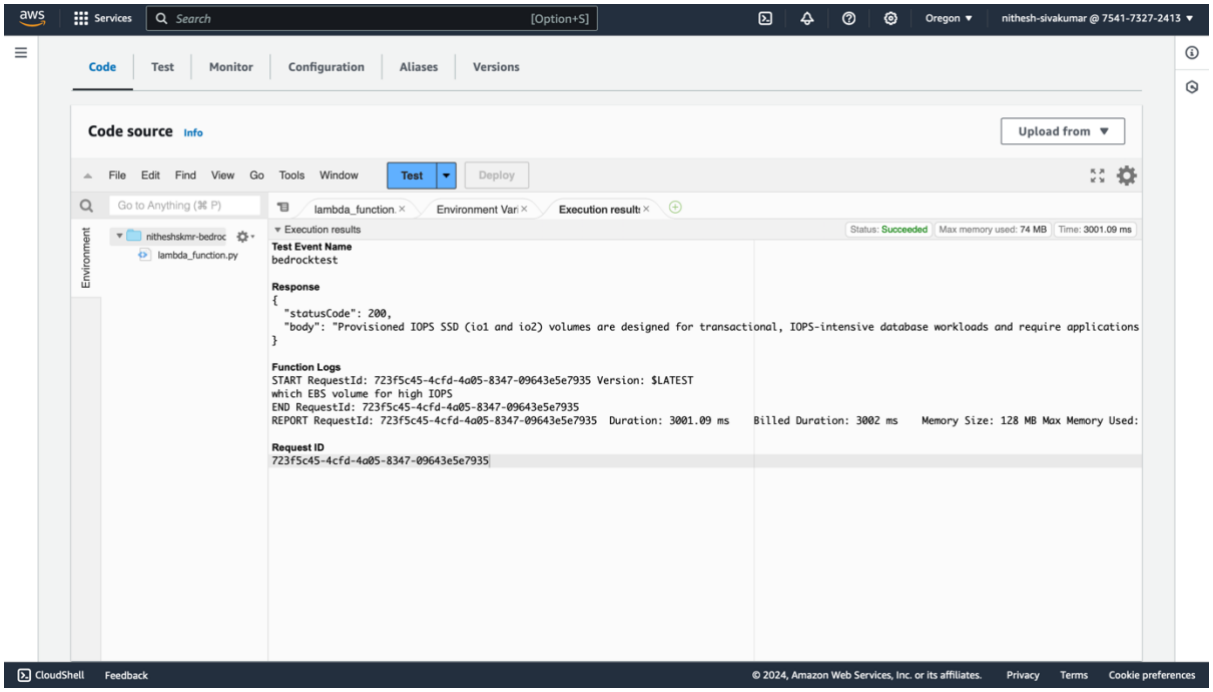
Run the test to check if the Lambda function retrieves the correct response from the knowledge base.



You can also view the **log events** in **Amazon CloudWatch** to monitor the execution details and debug any issues.



We can see a **200 status code**, indicating a successful request, along with the response from the knowledge base.



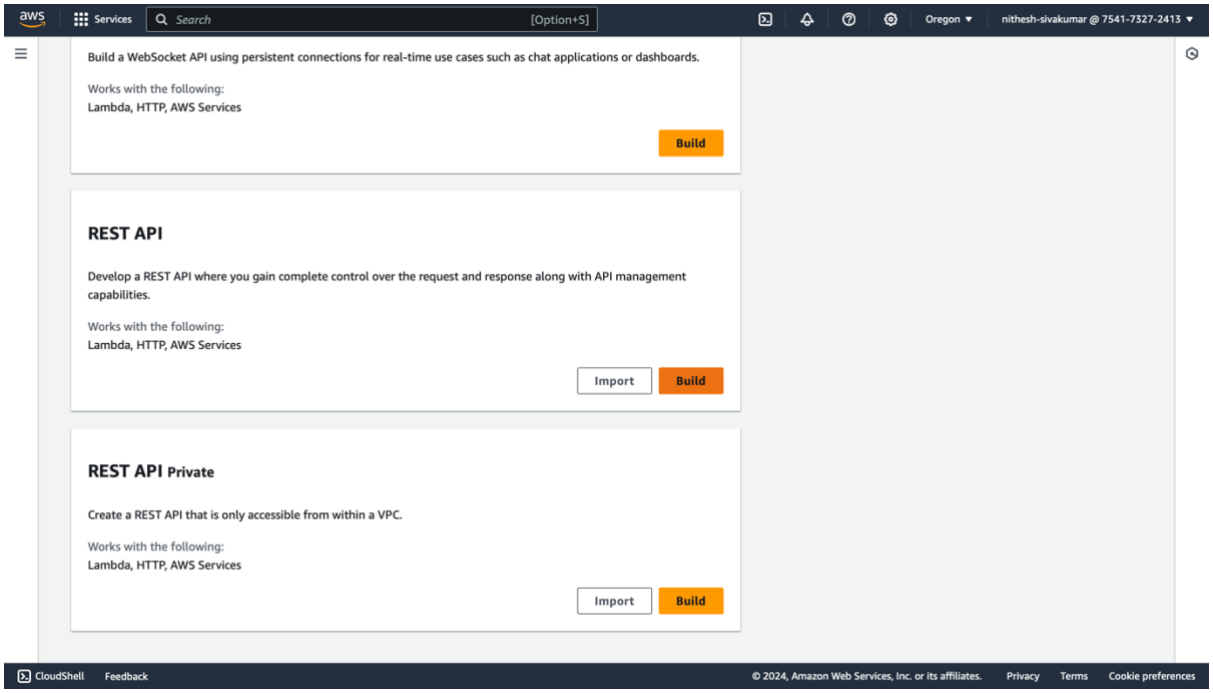
Creating an API Gateway

To expose the Lambda function as an endpoint, create an API Gateway.

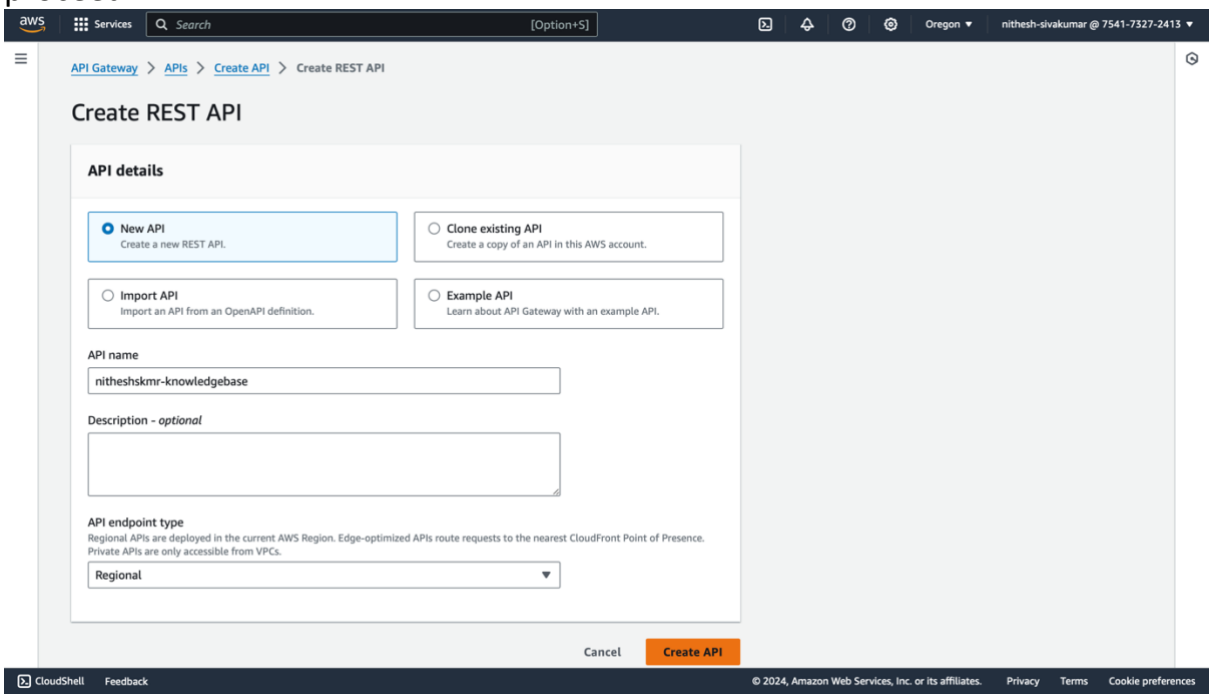
1. Search for **API Gateway** in the AWS console.
2. Click **Create API** and select **REST API**.

Why use API Gateway?

- Acts as an **entry point** for client requests.
- Handles **authentication, authorization, and rate limiting**.
- Allows secure and scalable communication between the frontend and backend services.
- Integrates seamlessly with **AWS Lambda**, enabling a **serverless API**.

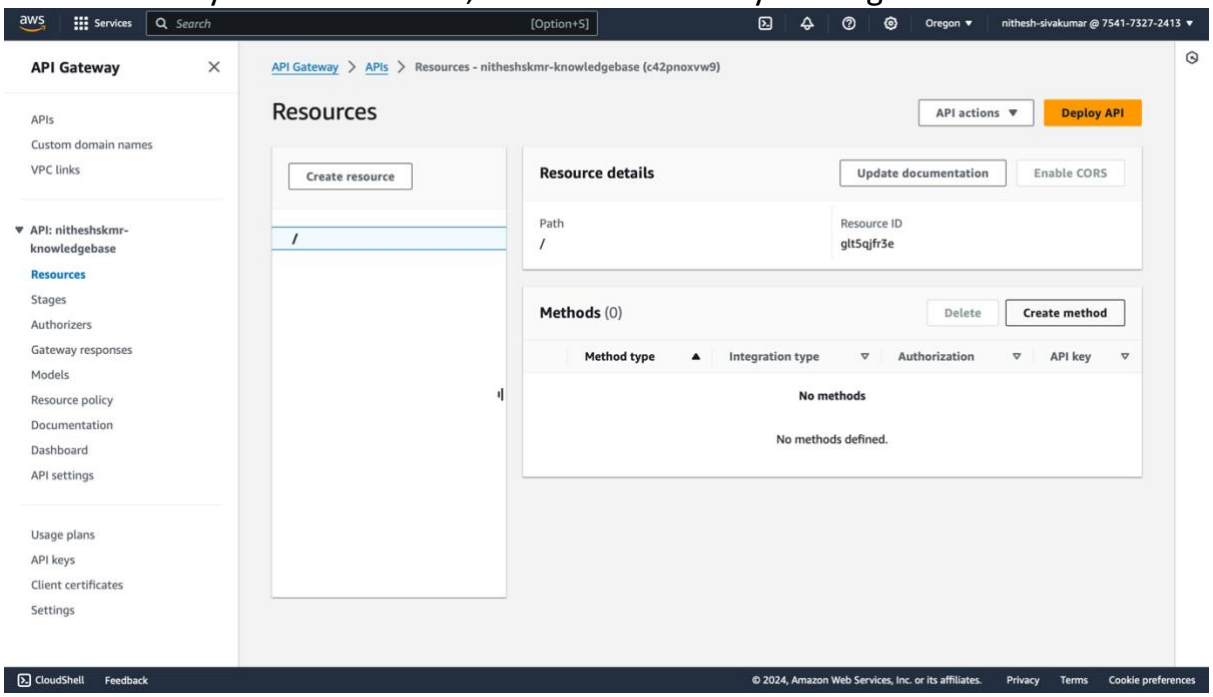


Give the API a name, such as **nitheshskmr-knowledgebase**, and click **Create** to proceed.

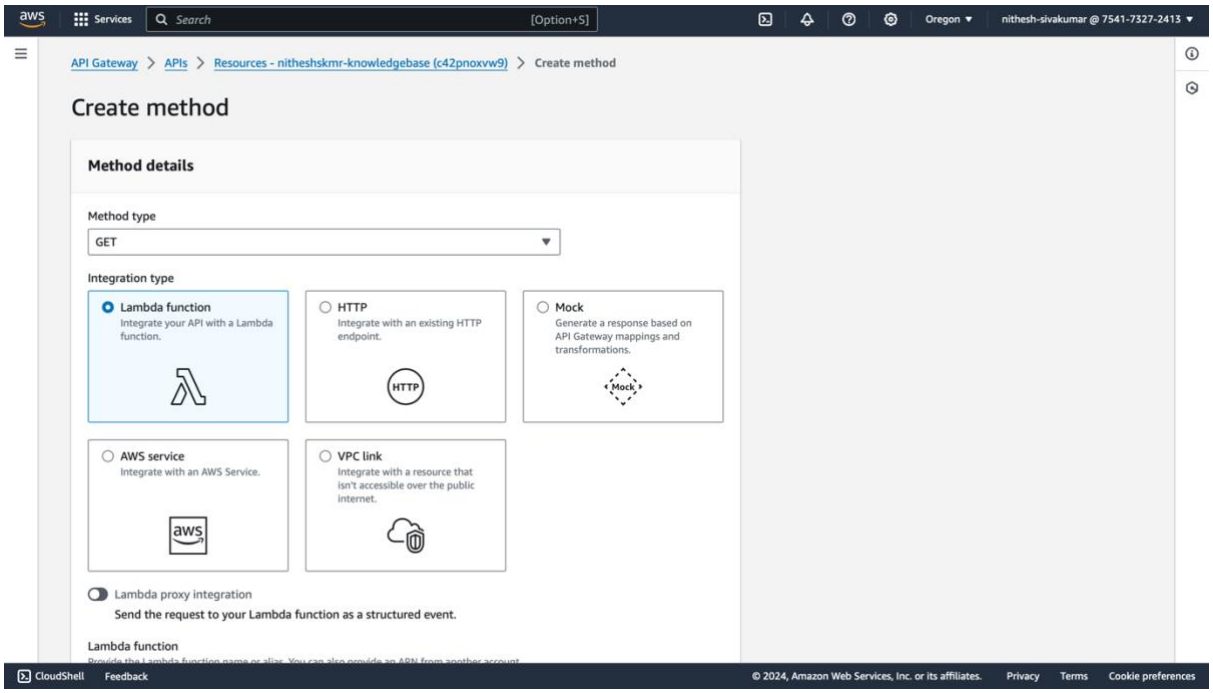


Creating a method in API Gateway defines how the API handles incoming requests. It specifies the **method** (GET, POST, etc.) and determines how requests are processed, including integration with AWS Lambda to retrieve responses from the knowledge base.

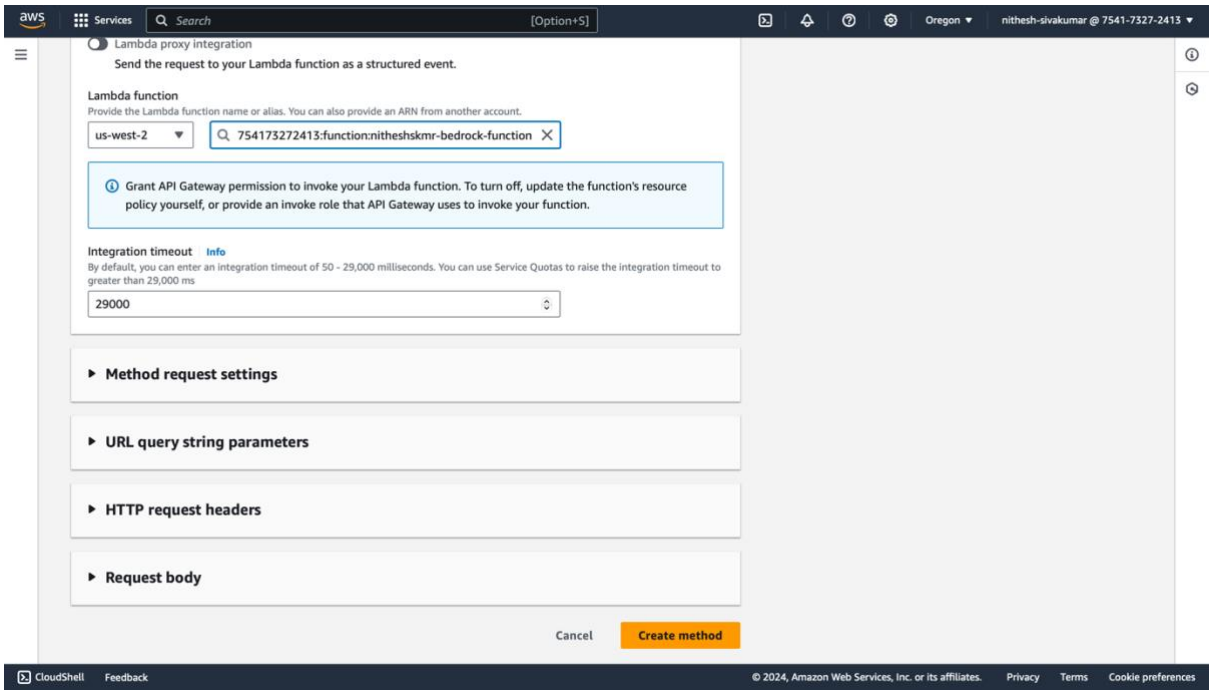
Go to the API you created. Now, create a method by clicking on **Create Method**.



Select **GET** as the method type and choose **Lambda Function** as the integration type. This allows API Gateway to forward GET requests to the Lambda function, which will process queries and return responses from the knowledge base.

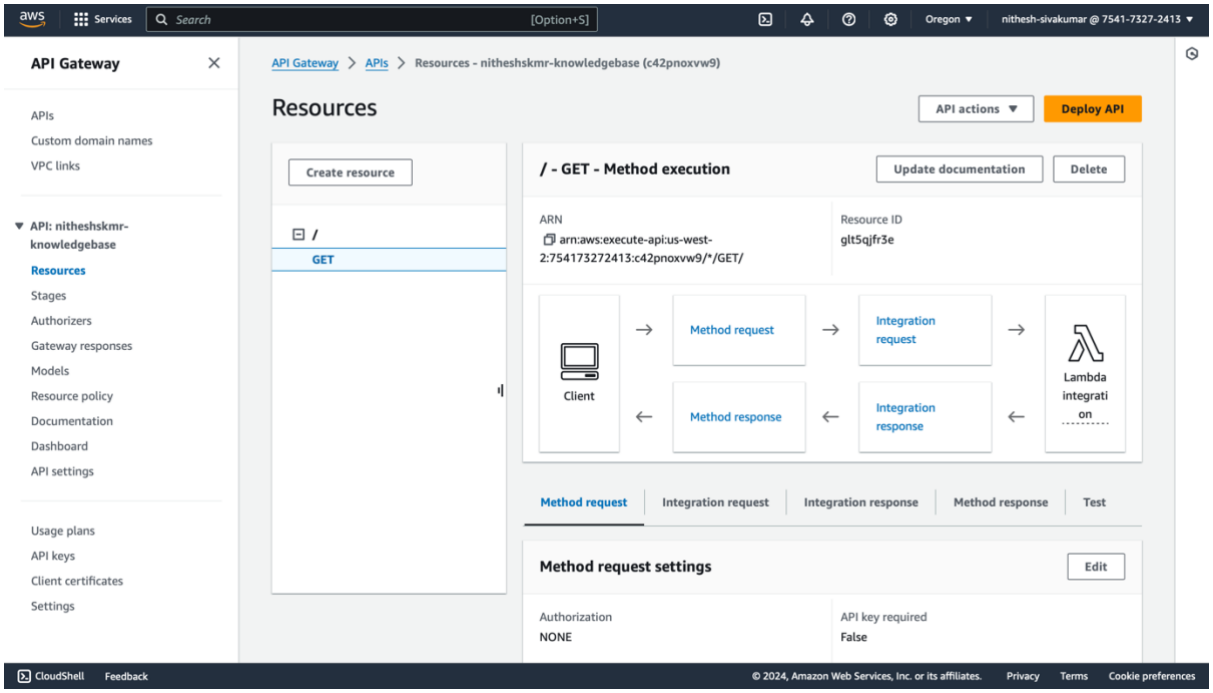


Select the **Lambda function** you created earlier, then click **Create Method** to complete the setup.



Configuring the Method Request

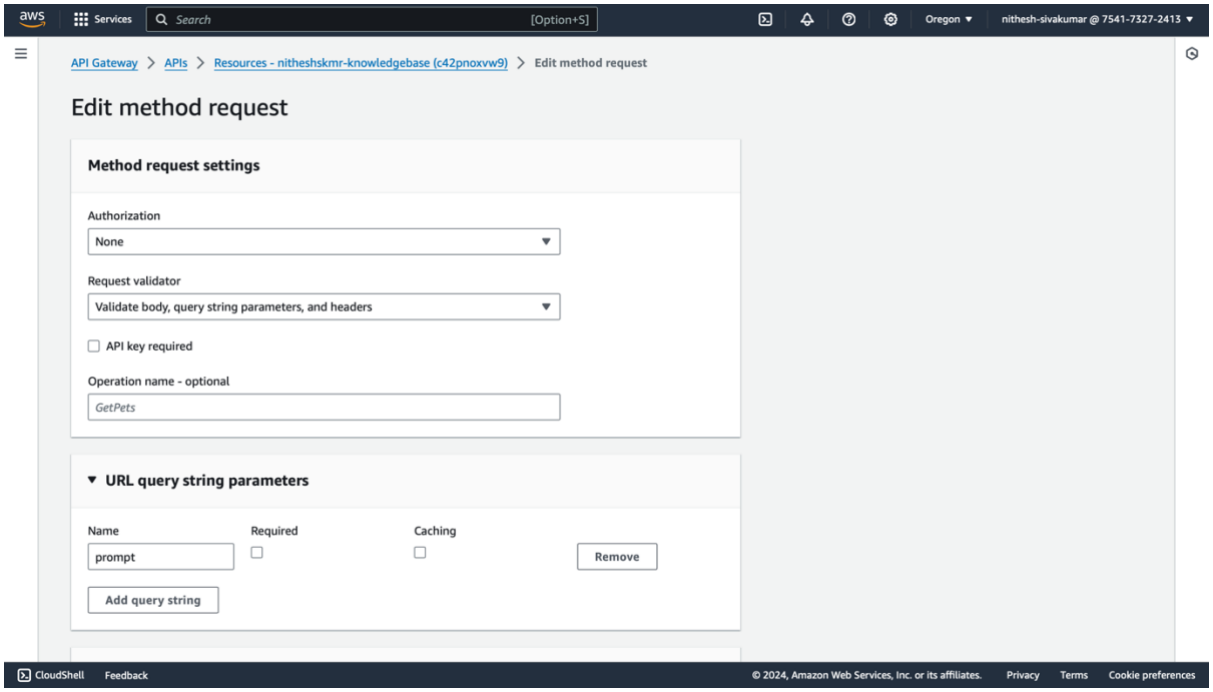
1. Select **Method Request** under the API method.
2. Click **Edit** to modify the settings.



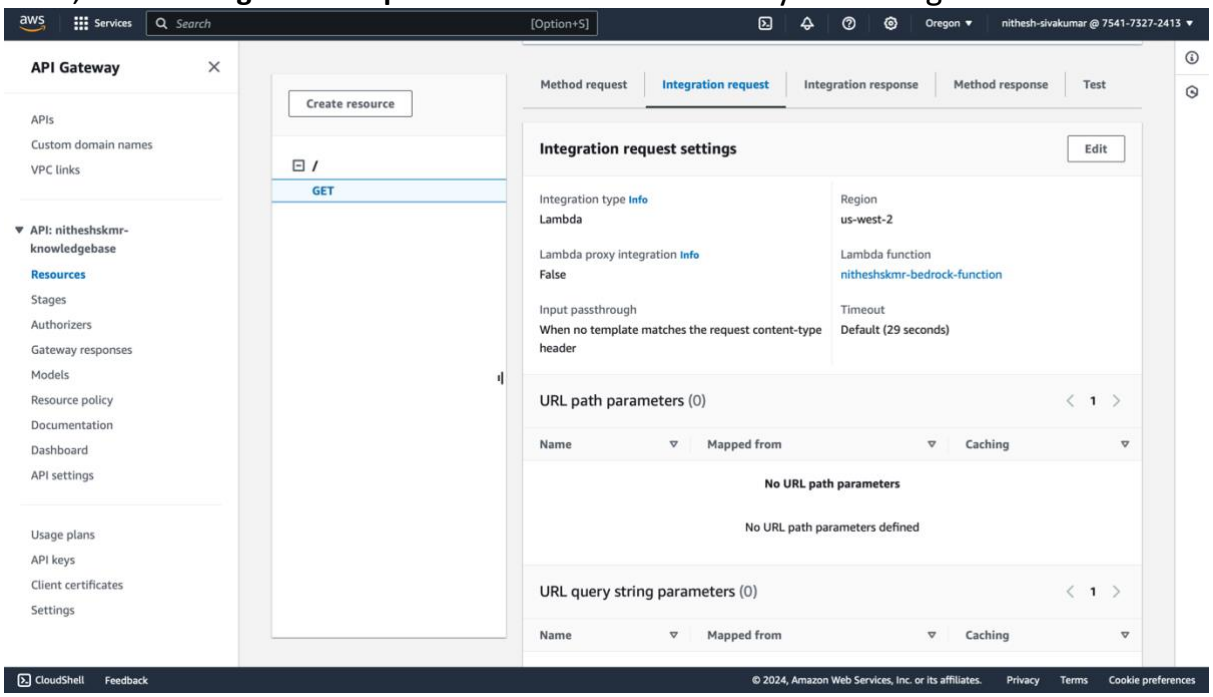
Steps to update settings:

- Enable **Request Validator** to ensure input validation.
- Click on **Request Validator** and select **Validate body, query string parameters, and headers**.
- Under **URL Query String Parameters**, click **Add Parameter** and enter the name as **prompt**. Click **Save**.

These changes ensure that API Gateway validates incoming requests and requires a query parameter for the prompt.



Next, select **Integration Request** and click **Edit** to modify its settings.

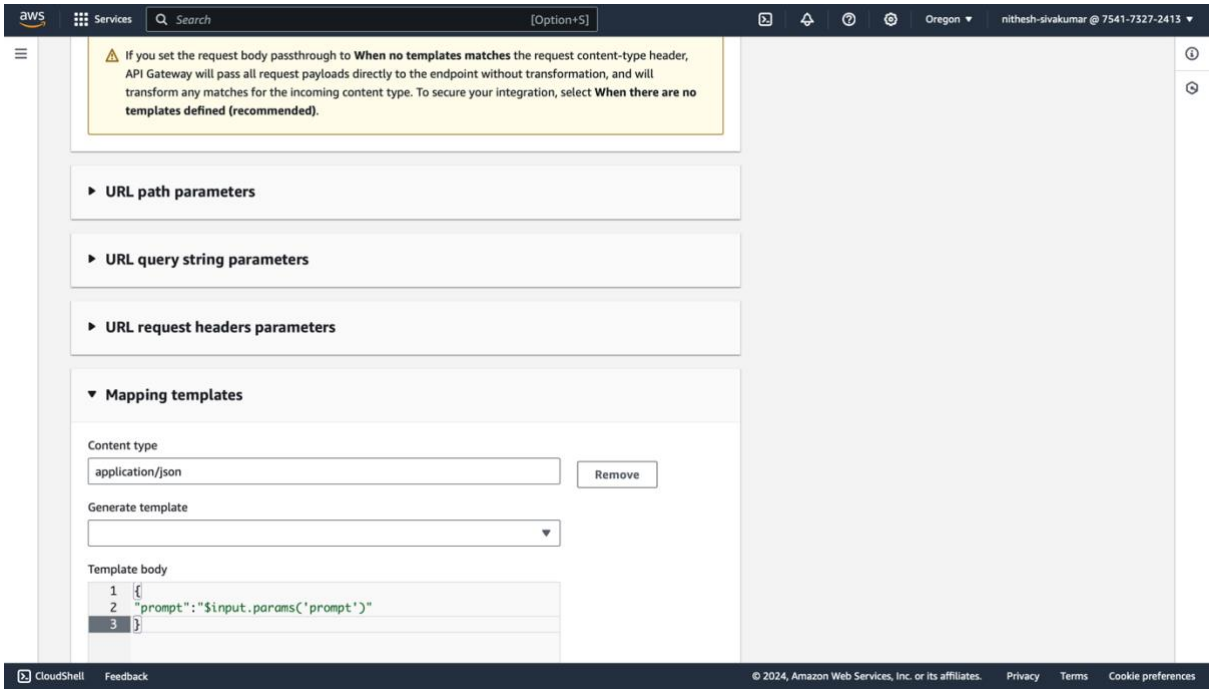


Scroll down to **Mapping Templates**, then click **Add**.

- Set **Content Type** to application/json.
- Under **Template Body**, copy and paste the provided JSON template.

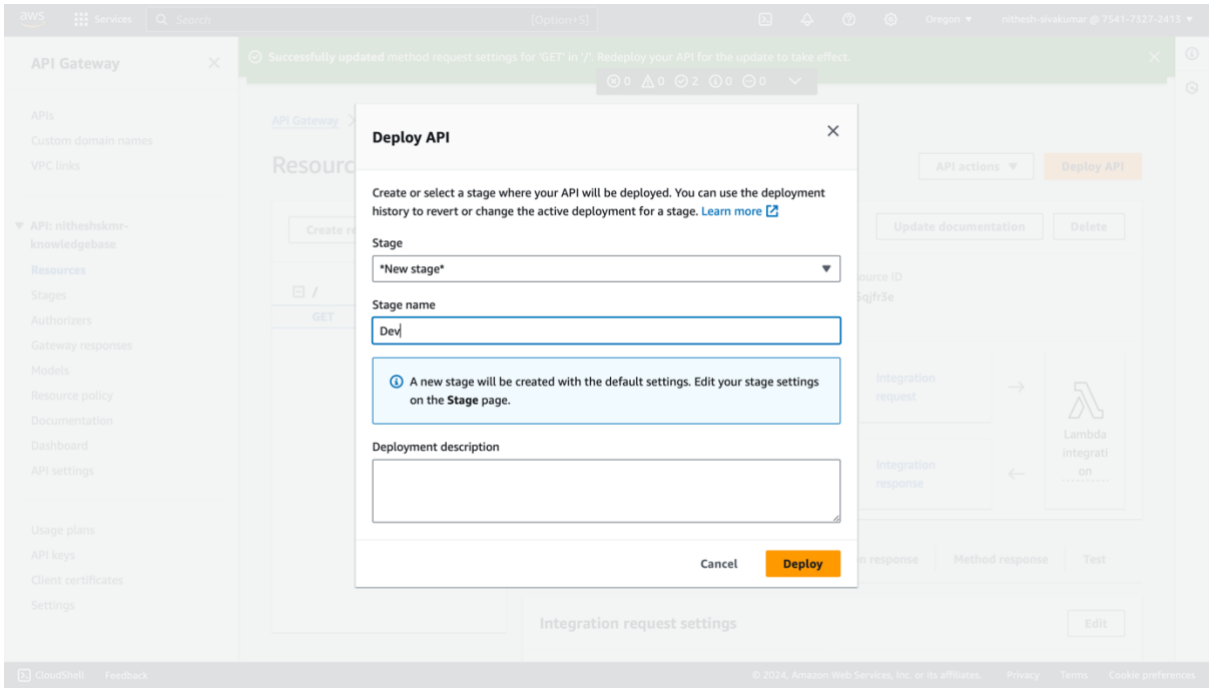
```
{

“prompt”:”$input.params(‘prompt’)”

}
```

This ensures that incoming API requests are properly formatted before being sent to the Lambda function.

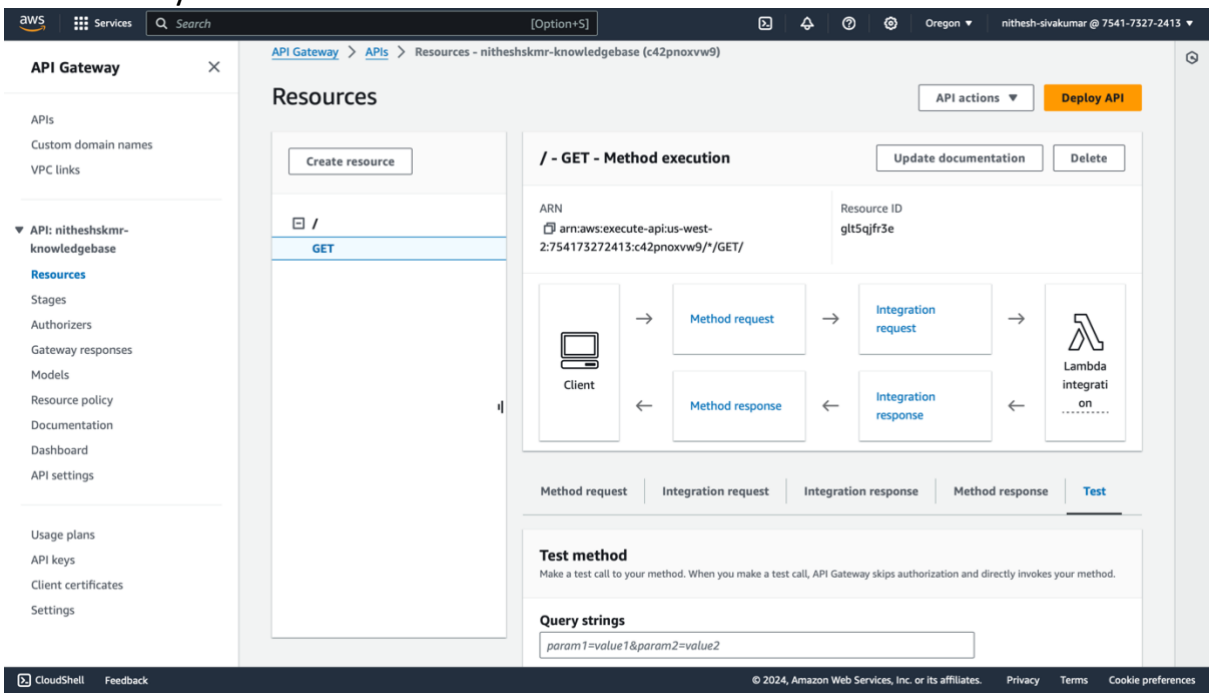


Next, deploy the API:

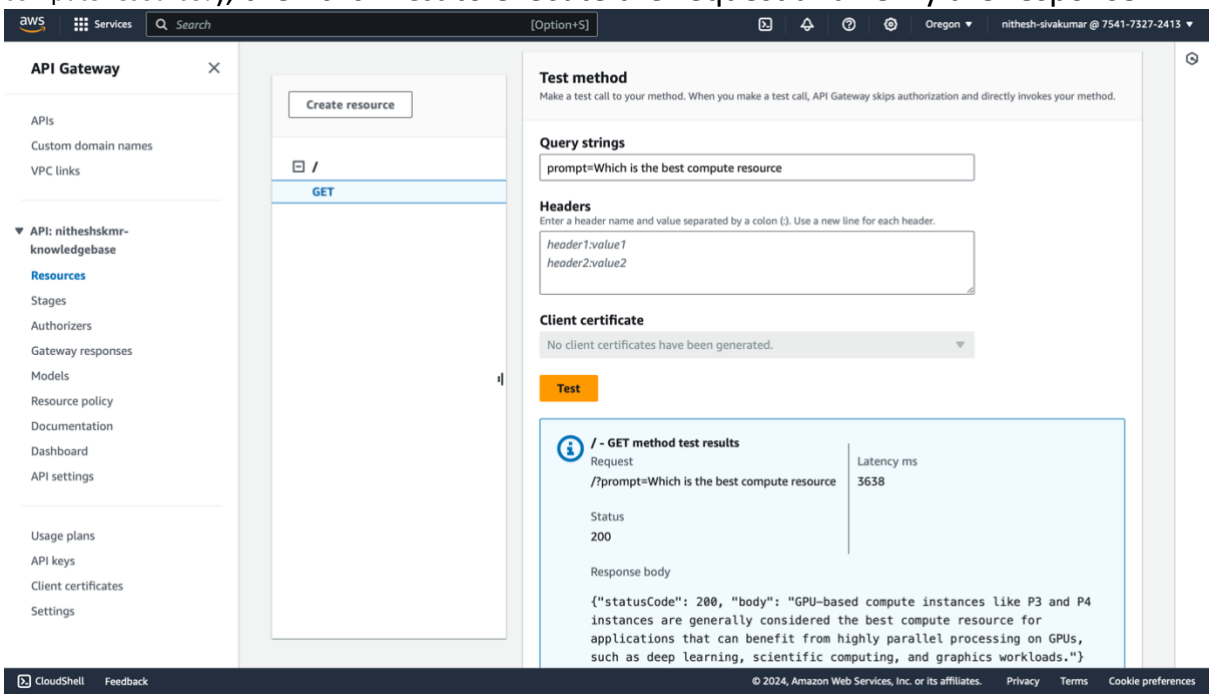
1. Click **Deploy API**.
2. Select **New Stage**.
3. Enter **Stage Name** as dev.
4. Click **Deploy** to make the API live.

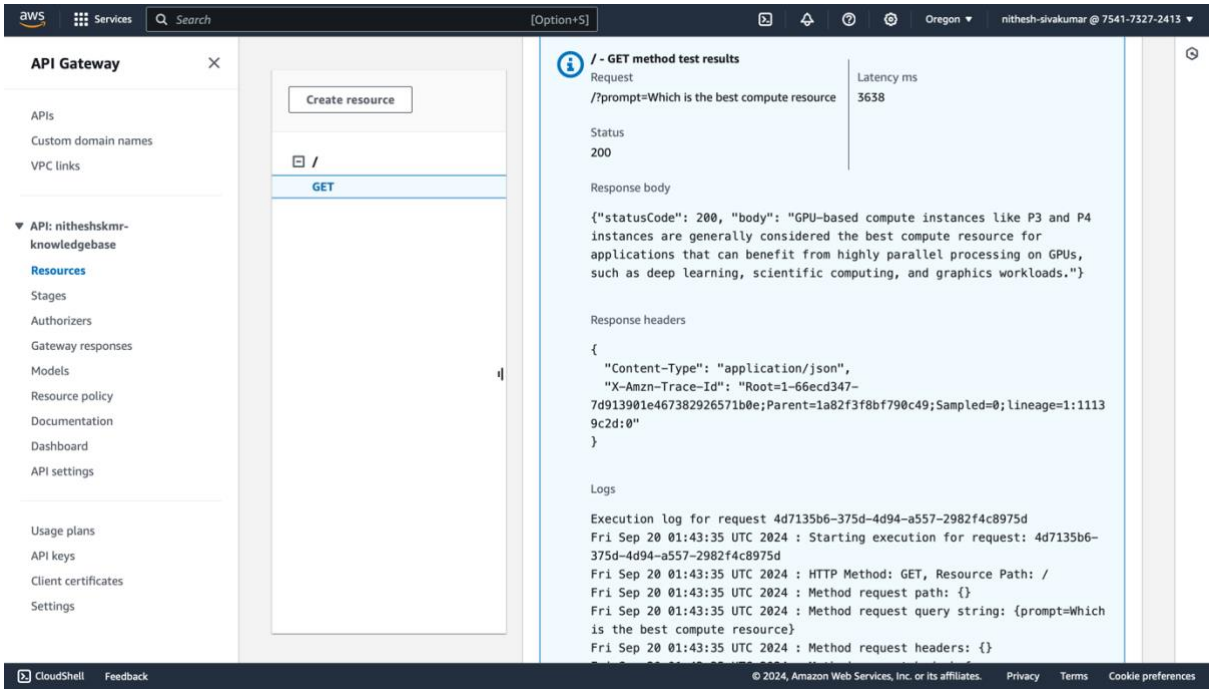


Next, select **Resources** from the left pane, then choose **Test** to verify the API's functionality.



Under **Query Strings**, enter the required parameters (e.g., prompt=Which is the best compute resource?), then click **Test** to execute the request and verify the response

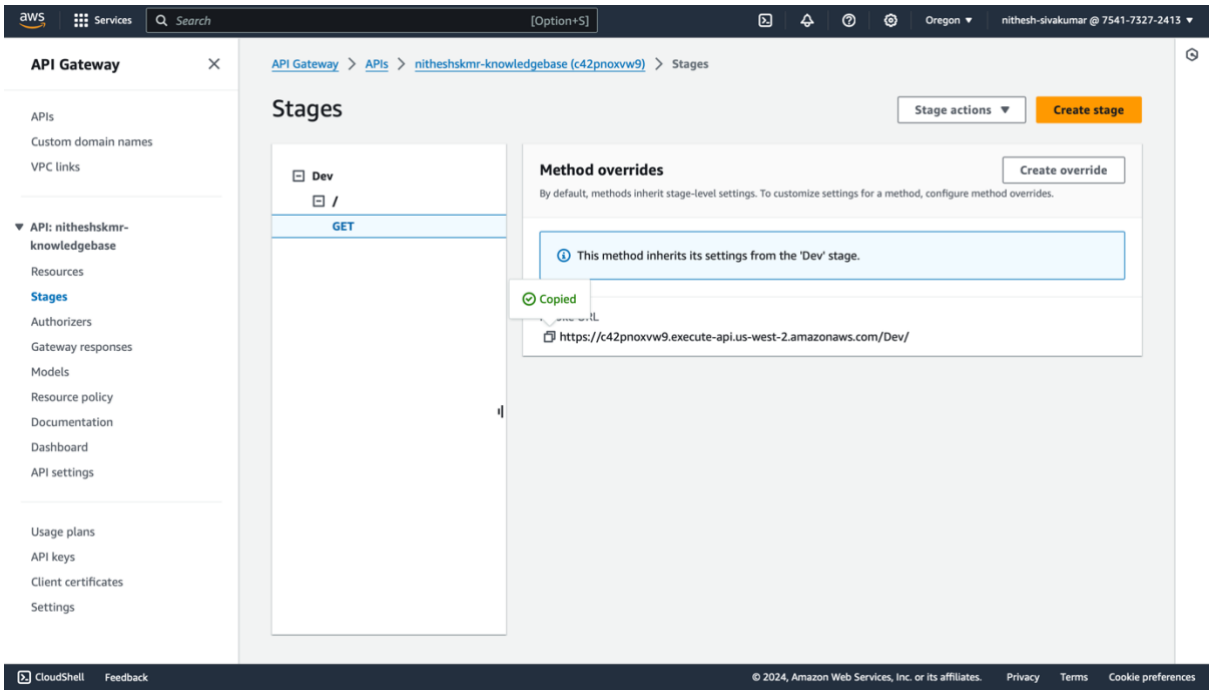


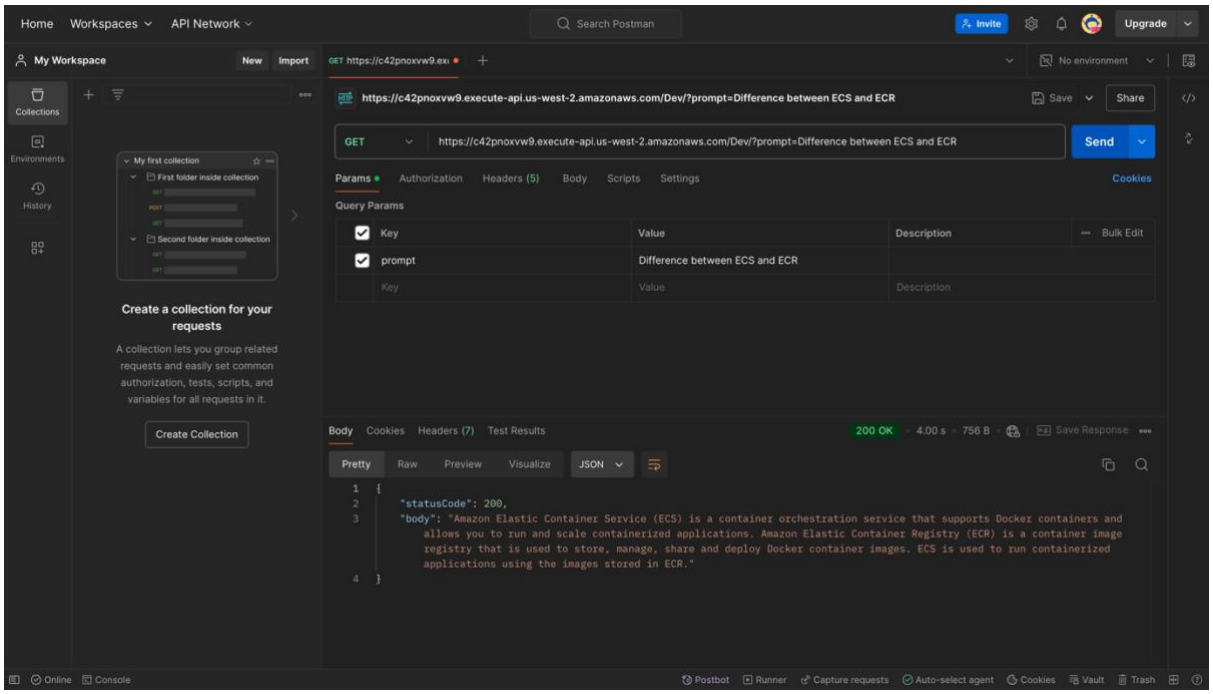


Success!

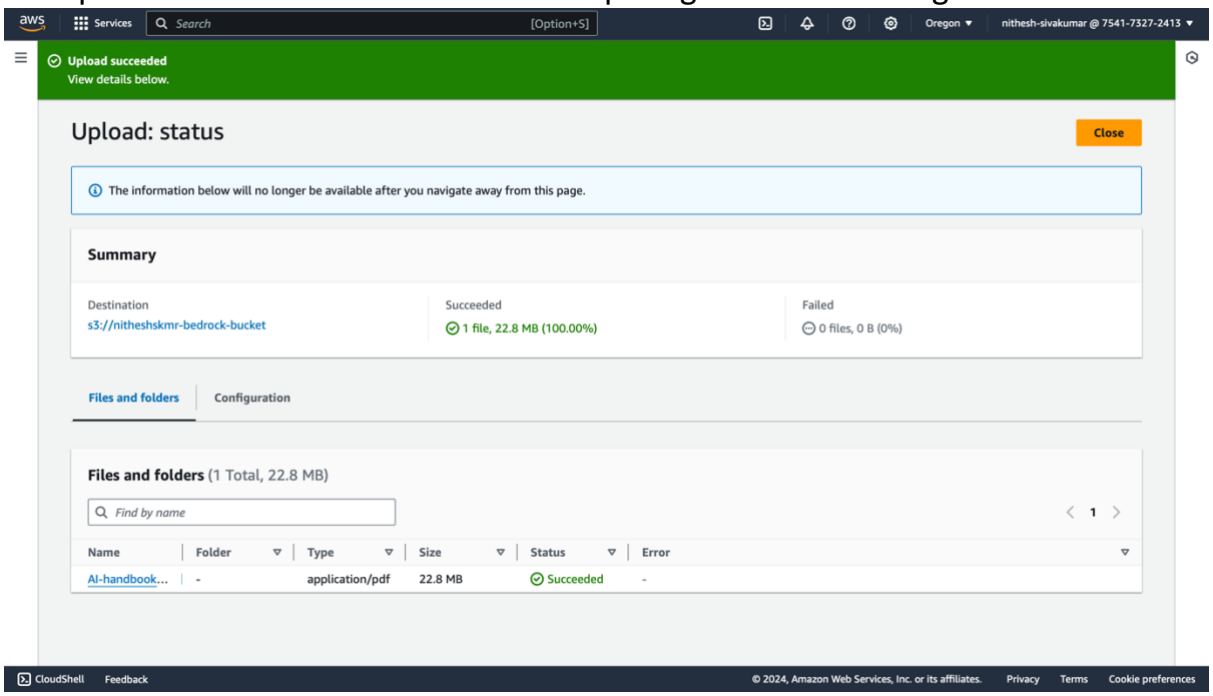
You can also test the API using **Postman**:

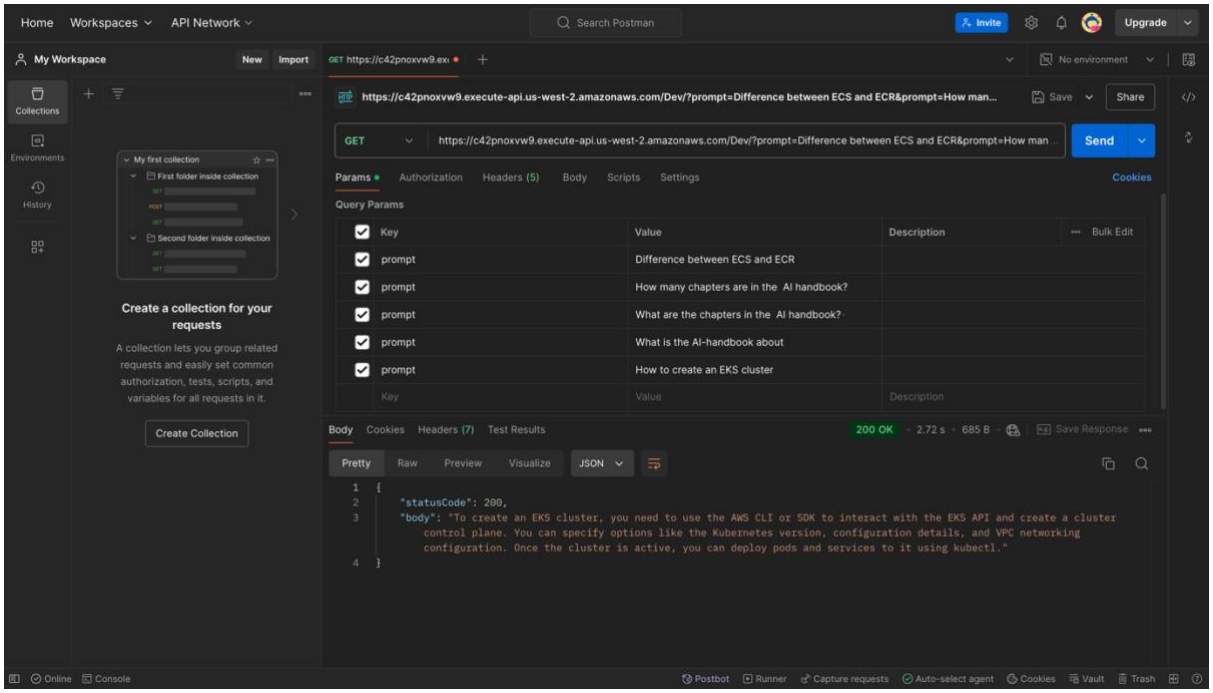
1. Copy the **API Invoke URL** from API Gateway.
2. Open **Postman**.
3. Paste the URL into the request field.
4. Set the **method** to **GET**.
5. Add the query parameter:
    - Key: prompt
    - Value: *Which EBS volume is best for high IOPS?*
6. Click **Send** to test the API response.

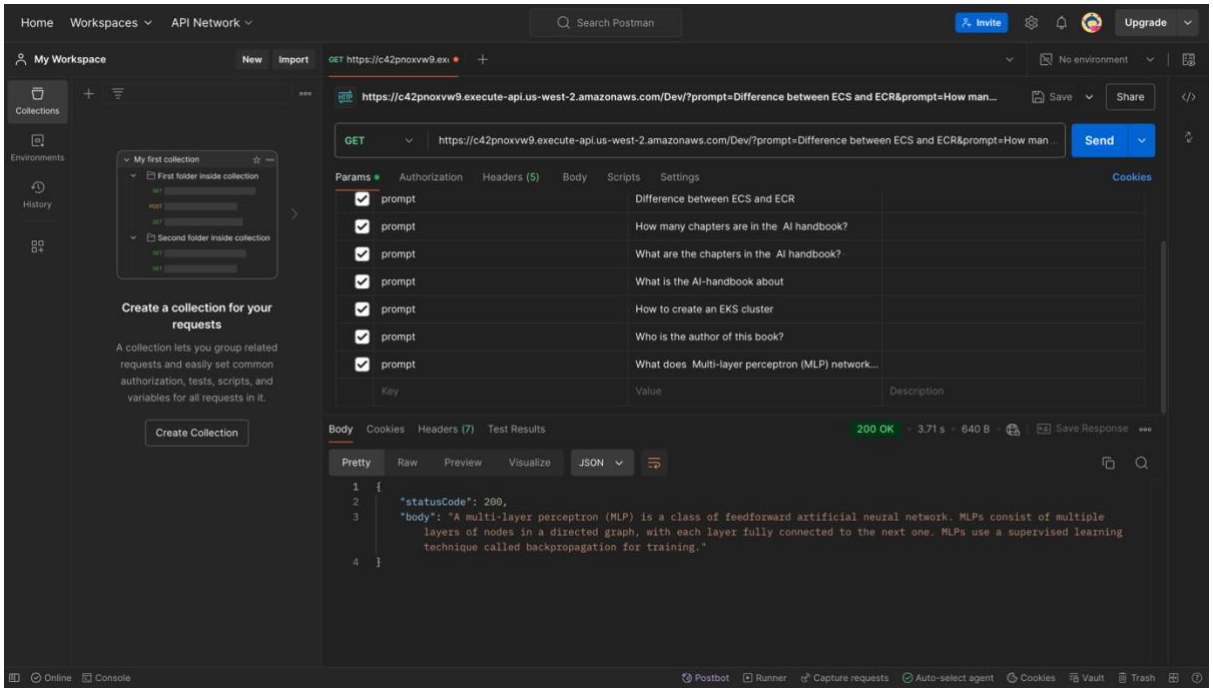




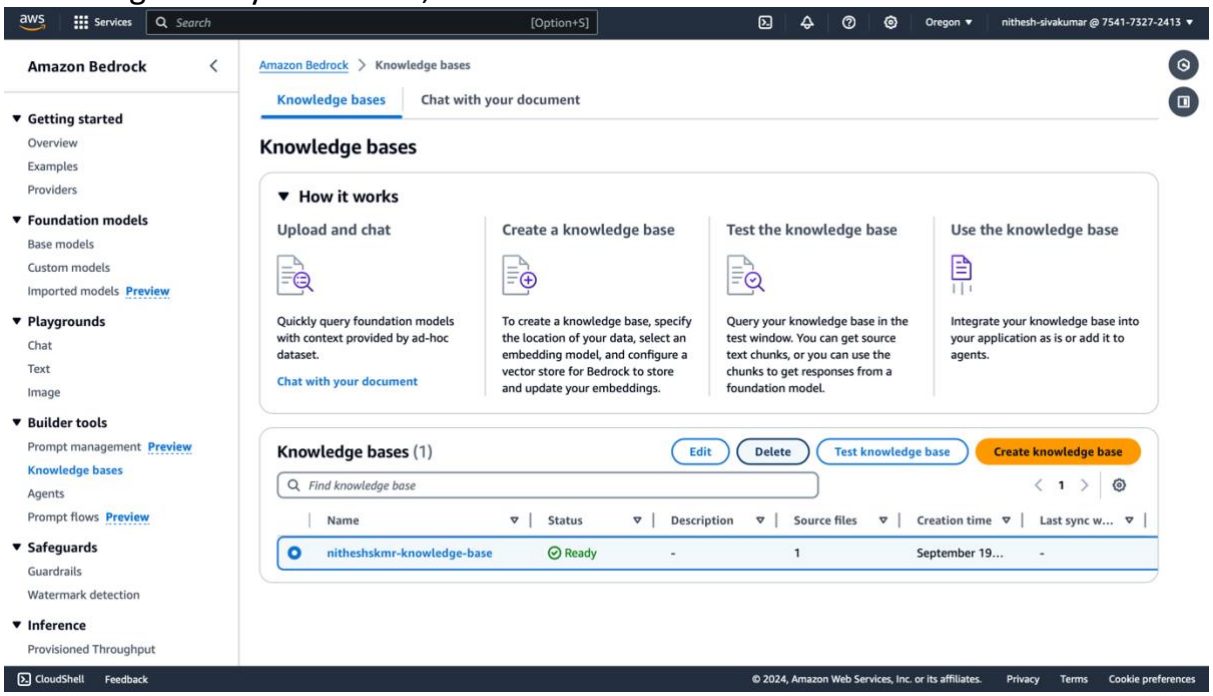
Uploaded a new PDF, **AI Handbook**, to the S3 bucket and tested the API again using **Postman**, and it successfully retrieved responses based on the updated knowledge base. This confirms that the system dynamically processes and incorporates new documents without requiring additional configuration.







Go to **Amazon Bedrock**, navigate to the **Knowledge Base** section, select the knowledge base you created, and click **Delete** to remove it.



Go to **Amazon OpenSearch**, locate the **vector database** created by the knowledge base, select it, and click **Delete** to remove it.

How the Vector Database Was Created

The vector database was not manually created. When setting up the **Knowledge Base** in Amazon Bedrock, selecting "Create New Vector Database" triggered AWS to automatically provision an **OpenSearch-managed vector database**. This database stores the vector embeddings of the documents uploaded to S3, enabling efficient retrieval of relevant information for user queries.