Steps First Create Lambda Function in AWS account

That lambda function will connect with AstraDB through code. In AstraDB we have loaded the required PDF files for vectorization and chunking that lambda function will have connection to DB details

In the below part highlighted in Yellow

import json

import urllib.request

def lambda\_handler(event, context):

    # Extract search term from event

    search\_term = event.get('search\_term', '')

    if not search\_term:

        return {

            'statusCode': 400,

            'body': json.dumps({'error': 'search\_term is required'})

        }

    # Hardcoded Astra DB configuration

    astra\_endpoint = "https://6336f9b3-93b6-4ff7-83ba-1aba47d9b905-us-east-2.apps.astra.datastax.com"

    astra\_token = "AstraCS:thsIOqmwPDdDKqGJYtxTRJsT:01b2f53a5b13f96788519df2ced3b48849240da00a81f9db882e10840fe81720"

    keyspace = "default\_keyspace"

    collection = "db2\_collection"

    # Construct API URL - corrected format

    api\_url = f"{astra\_endpoint}/api/json/v1/{keyspace}/{collection}"

    # Vector similarity search query

    query\_data = {

        "find": {

            "sort": {

                "$vectorize": search\_term

            },

            "options": {

                "limit": 5,

                "includeSimilarity": True

            }

        }

    }

    try:

        # Prepare request

        headers = {

            'Content-Type': 'application/json',

            'Token': astra\_token  # Changed from X-Cassandra-Token

        }

        # Make HTTP request

        req = urllib.request.Request(

            api\_url,

            data=json.dumps(query\_data).encode('utf-8'),

            headers=headers,

            method='POST'

        )

        # Execute request with error handling

        try:

            with urllib.request.urlopen(req, timeout=30) as response:

                result = json.loads(response.read().decode('utf-8'))

        except urllib.error.HTTPError as e:

            error\_body = e.read().decode('utf-8')

            return {

                'statusCode': e.code,

                'body': json.dumps({

                    'error': f'Astra DB API Error: {e.code}',

                    'details': error\_body

                })

            }

        # Extract results

        clean\_results = []

        if 'data' in result and 'documents' in result['data']:

            for doc in result['data']['documents']:

                # Include both the vectorize content and similarity score

                result\_item = {}

                if '$vectorize' in doc:

                    result\_item['content'] = doc['$vectorize']

                if '$similarity' in doc:

                    result\_item['similarity'] = doc['$similarity']

                # Include other fields if present

                for key, value in doc.items():

                    if not key.startswith('$'):

                        result\_item[key] = value

                clean\_results.append(result\_item)

        return {

            'statusCode': 200,

            'body': json.dumps({

                'search\_term': search\_term,

                'results': clean\_results,

                'count': len(clean\_results)

            })

        }

    except urllib.error.URLError as e:

        return {

            'statusCode': 500,

            'body': json.dumps({

                'error': 'Network error',

                'details': str(e.reason)

            })

        }

    except Exception as e:

        return {

            'statusCode': 500,

            'body': json.dumps({

                'error': 'Unexpected error',

                'details': str(e),

                'type': type(e).\_\_name\_\_

            })

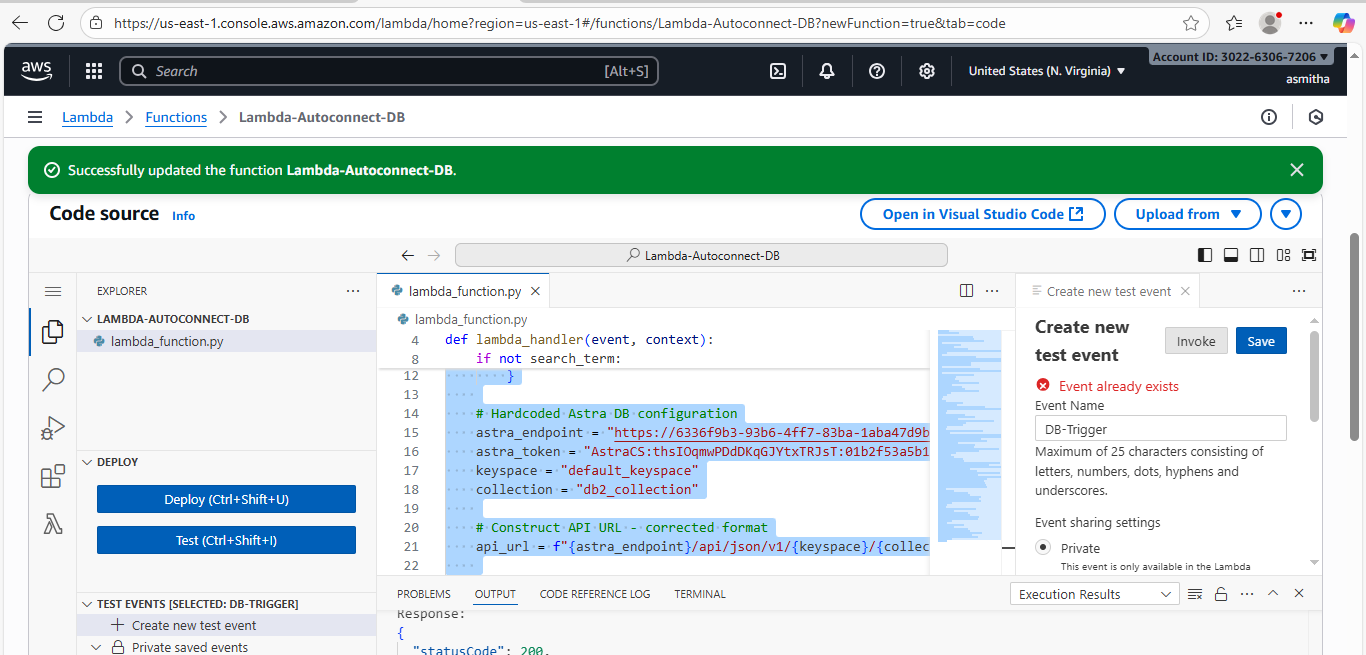
        }

In AstraDB we should give all the collection and token Details

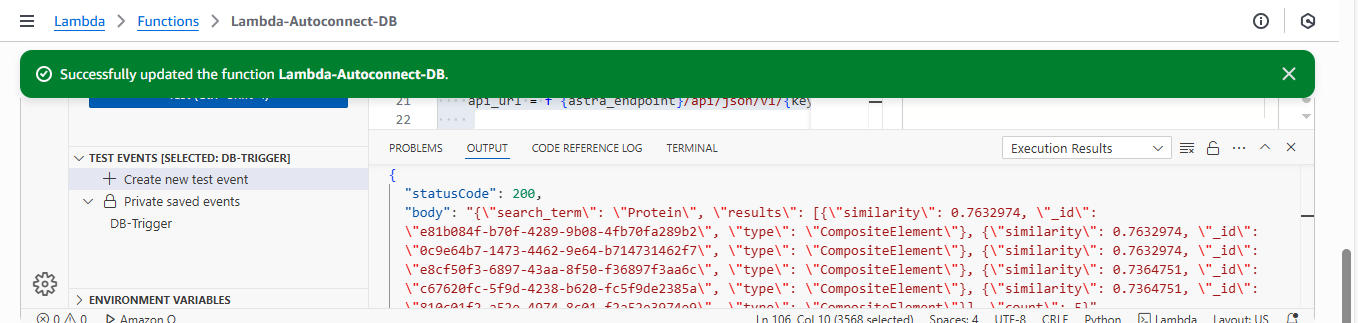
We have already Loaded PDF in AstraDB

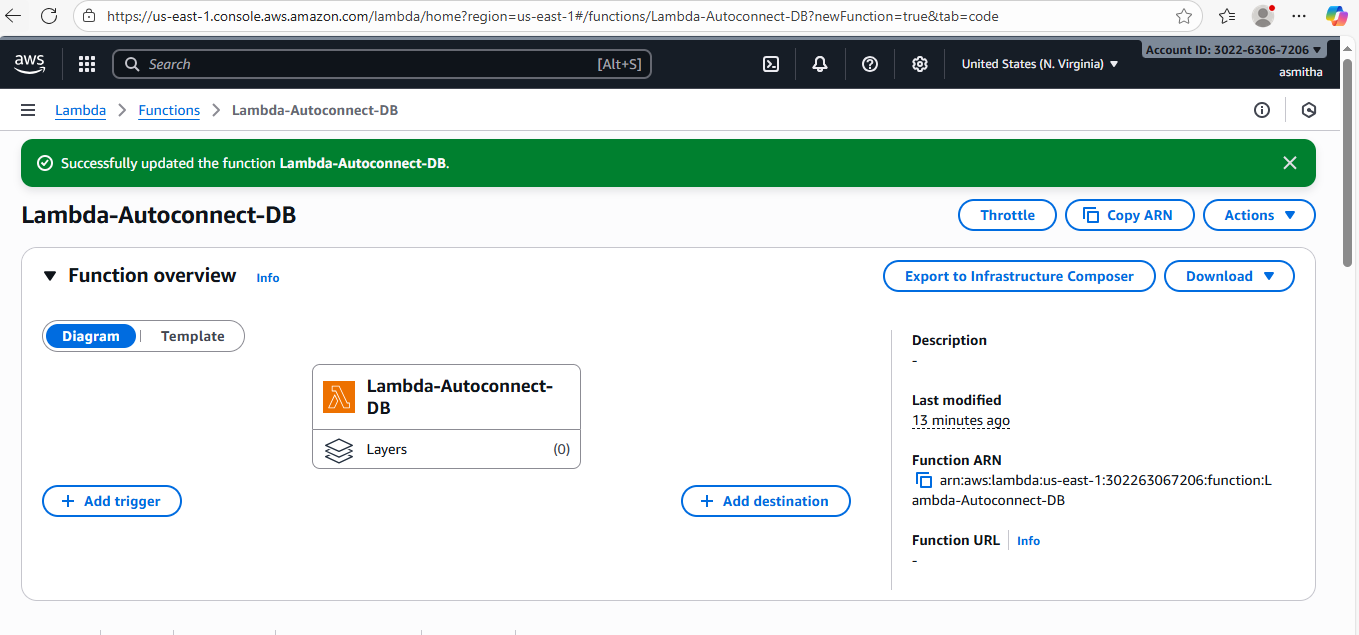
Create Lambda Function

And also create event to trigger the function



Protein is the content u want to search in Nutrition PDF





This is function ARN : arn:aws:lambda:us-east-1:302263067206:function:Lambda-Autoconnect-DB

Now take the above code of lambda function in claude.ai and use the below prompt

Create an openapi 3.0 schema to help my bedrock agent invoke the above lambda accurately

This Prompt will create OPENAPI Schema for of lambda function which will be passed in BedROCK Agent option

openapi: 3.0.0

info:

title: Astra DB Vector Search API

description: API for performing semantic vector similarity search in Astra DB using natural language queries

version: 1.0.0

paths:

/search:

post:

summary: Search for semantically similar content

description: Performs a vector similarity search in Astra DB to find documents that are semantically similar to the provided search term. Returns the top 5 most relevant results with similarity scores.

operationId: searchAstraDB

requestBody:

required: true

content:

application/json:

schema:

type: object

required:

- search\_term

properties:

search\_term:

type: string

description: The natural language search query or phrase to find semantically similar content

example: "machine learning algorithms"

minLength: 1

maxLength: 1000

responses:

'200':

description: Successful search operation

content:

application/json:

schema:

type: object

properties:

search\_term:

type: string

description: The original search term that was queried

results:

type: array

description: Array of search results ordered by similarity (most similar first)

items:

type: object

properties:

content:

type: string

description: The text content of the matched document

similarity:

type: number

format: float

description: Similarity score between 0 and 1, where 1 is most similar

minimum: 0

maximum: 1

count:

type: integer

description: Number of results returned

minimum: 0

maximum: 5

example:

search\_term: "machine learning algorithms"

results:

- content: "Neural networks are a type of machine learning algorithm inspired by the human brain"

similarity: 0.89

- content: "Deep learning is a subset of machine learning that uses multiple layers"

similarity: 0.85

- content: "Random forests are ensemble learning algorithms used for classification"

similarity: 0.78

count: 3

'400':

description: Bad request - search\_term is missing or invalid

content:

application/json:

schema:

type: object

properties:

error:

type: string

description: Error message describing what went wrong

example:

error: "search\_term is required"

'500':

description: Internal server error - database connection or query failed

content:

application/json:

schema:

type: object

properties:

error:

type: string

description: High-level error description

details:

type: string

description: Detailed error information for debugging

type:

type: string

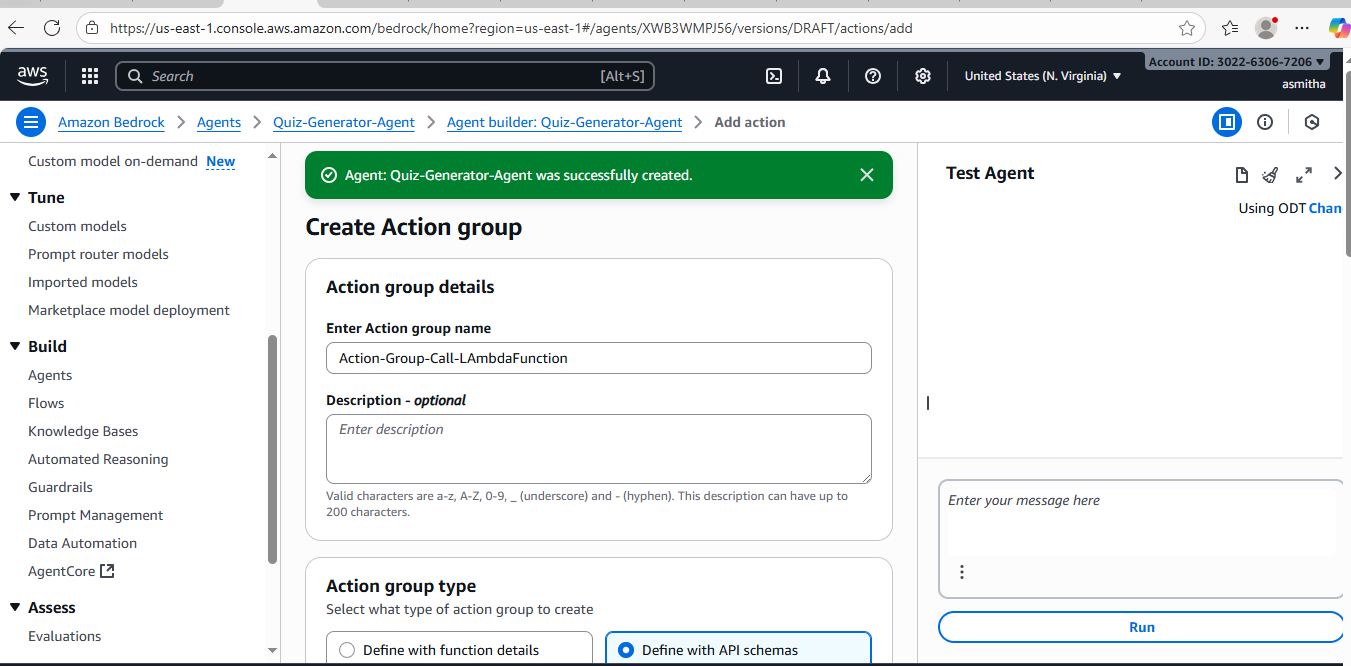
description: Type of exception that occurred

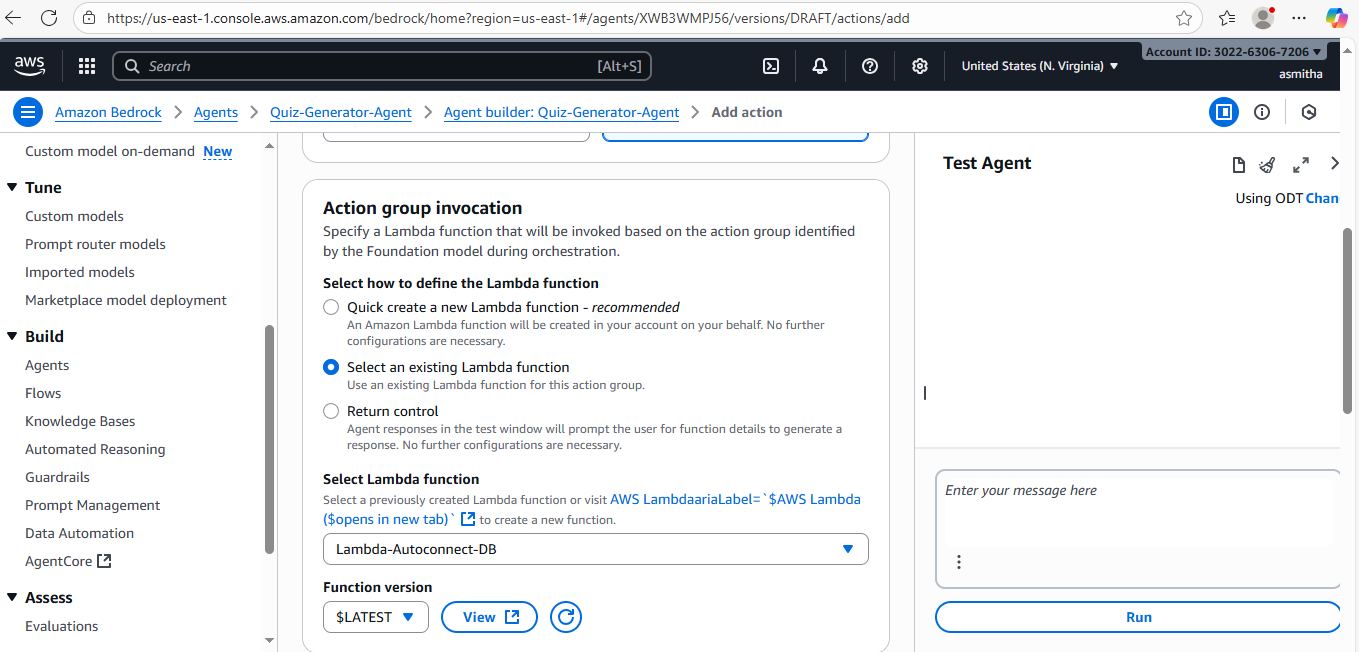
example:

error: "Astra DB API Error: 500"

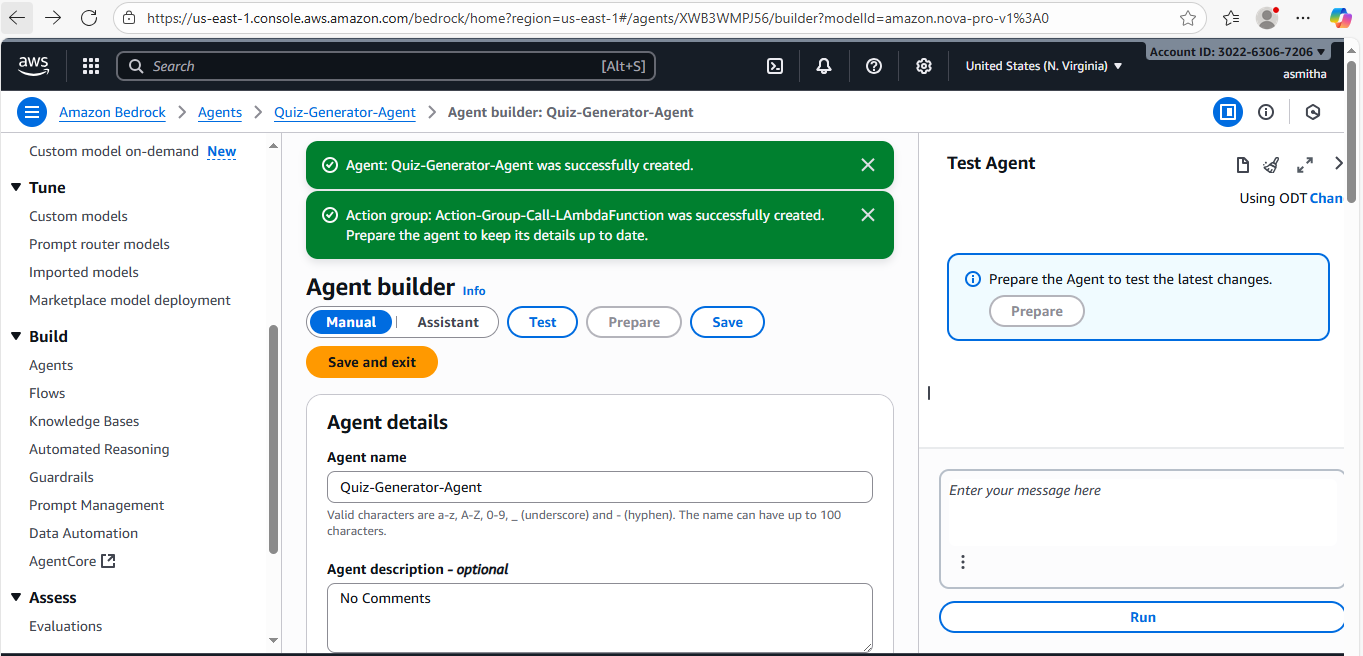
details: "Internal database error"

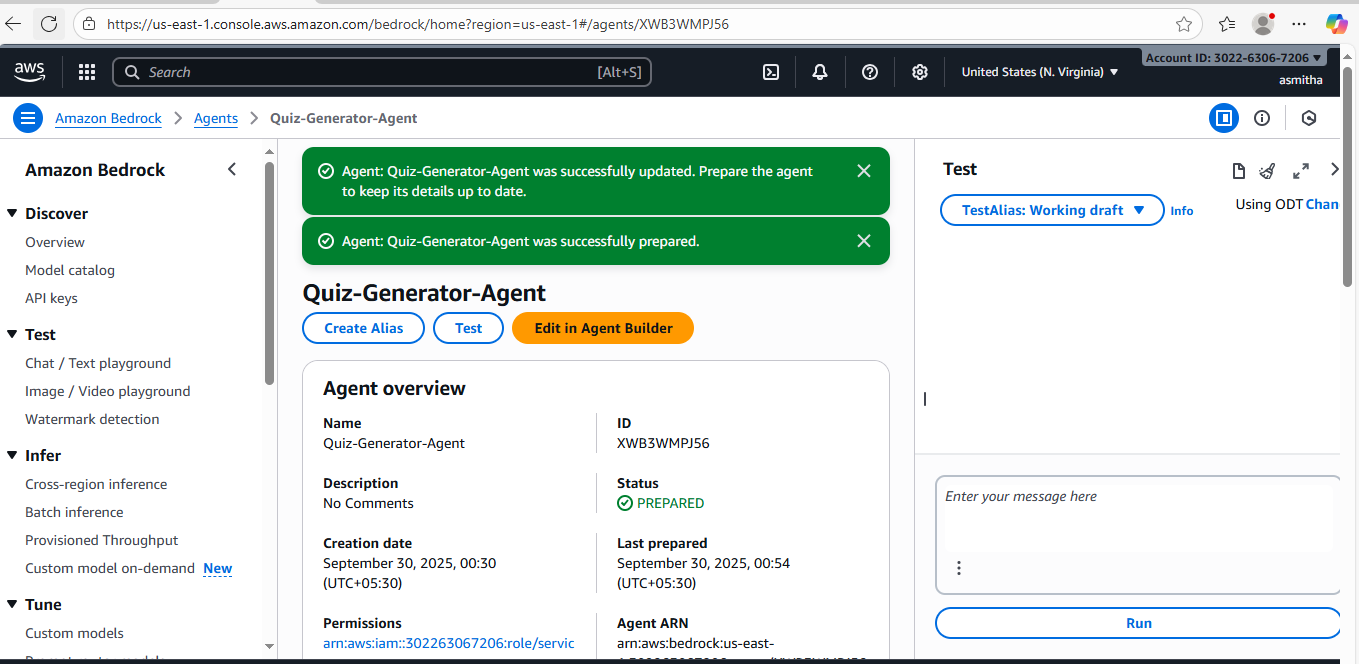
This way we can create action group by adding the OPENAPI Schema and selecting respective Lambda Function

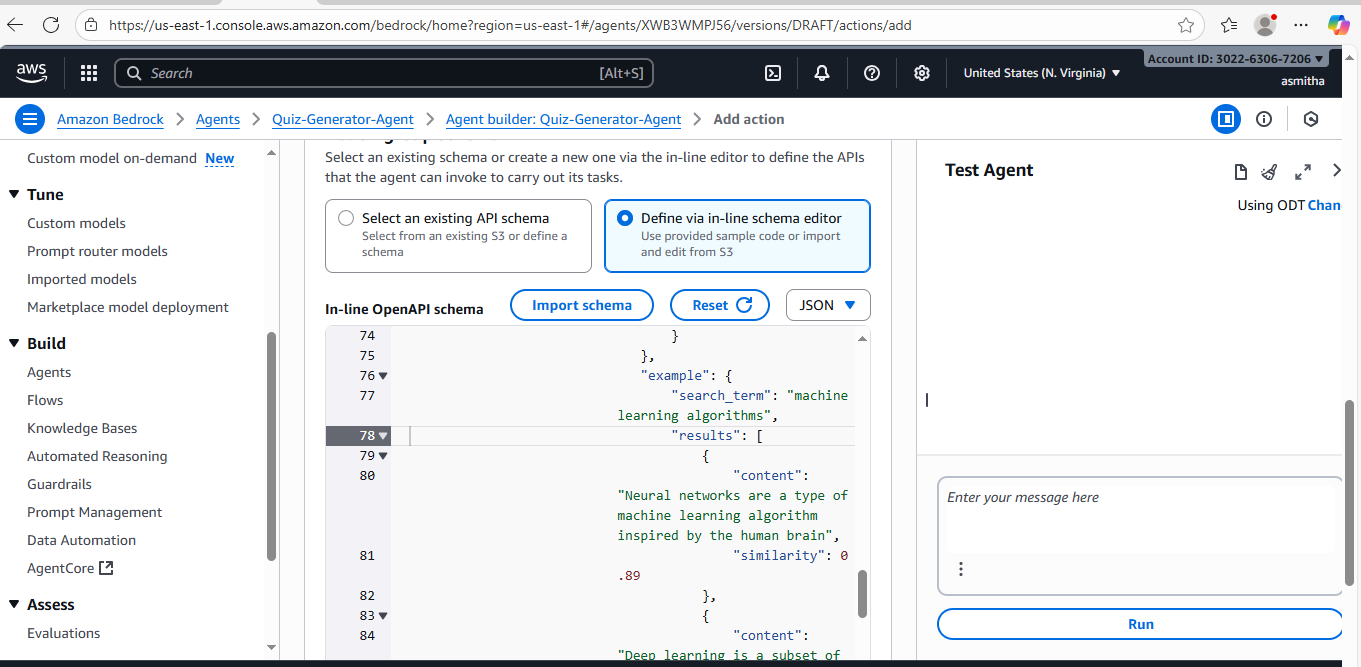




Now Prepare the Agent to test the Latest Changes







This schema helps the Lambda function to understand which parameter values match according to Key-Value pair

In the Below way we can create Flows in BedRock where we can call the respective Agents which has the respective lambda function in action group to invoke astra DB

