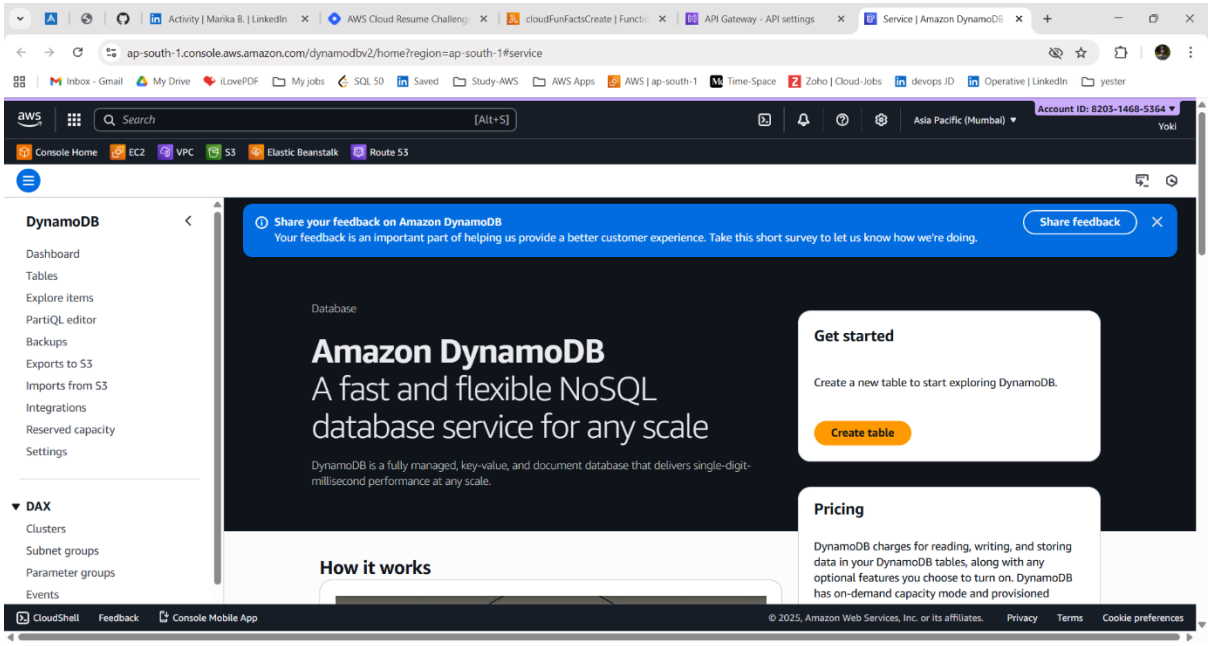
2 - Database Version (DynamoDB + Lambda)

~ In this 2nd stage, you will store cloud fun facts in a **DynamoDB Table** and fetch data dynamically.

~ This makes your API flexible, scalable, and production-ready.



Step - 1: Create DynamoDB Table:

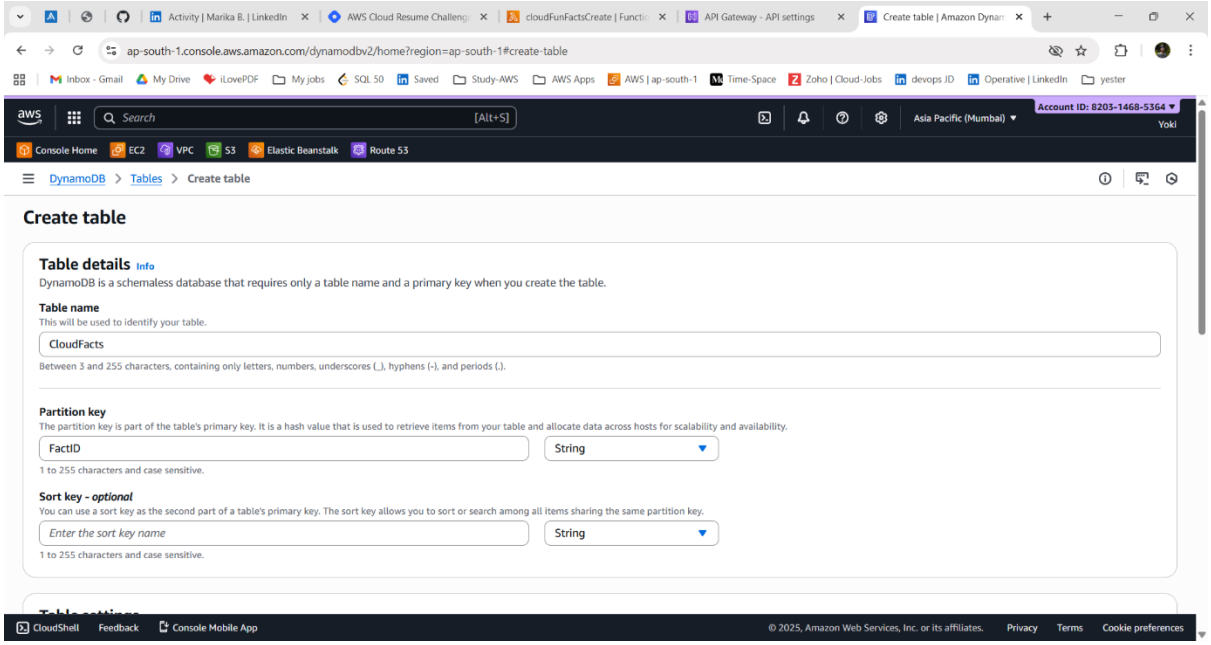
1. Go to AWS Management Console → Search for **DynamoDB.**

2. click "**Create Table**"

3. give table details.

    Table name: CloudFacts

    Partition key: FactID - (String)



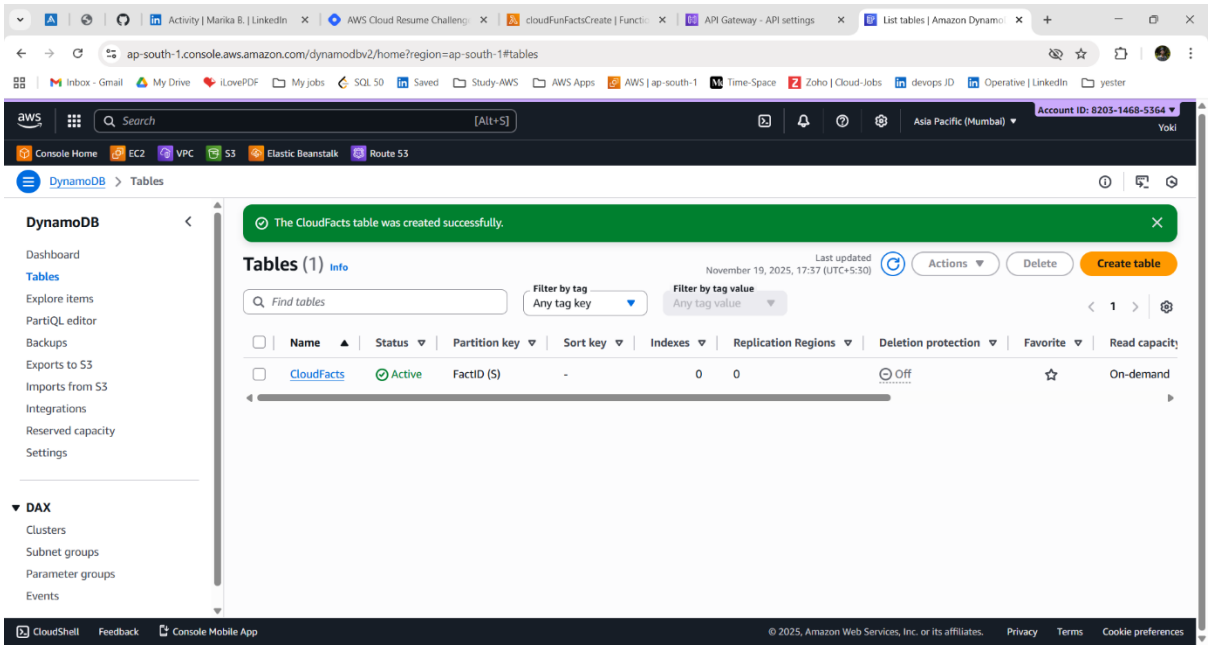
~ Leave other settings default (on-demand capacity is fine for this project).

👉 What's happening here?

- DynamoDB is **AWS's NoSQL database** - super fast, serverless, and fully managed.
- We're creating a table where each fact has:

  FactID → unique identifier (like 1, 2, 3)

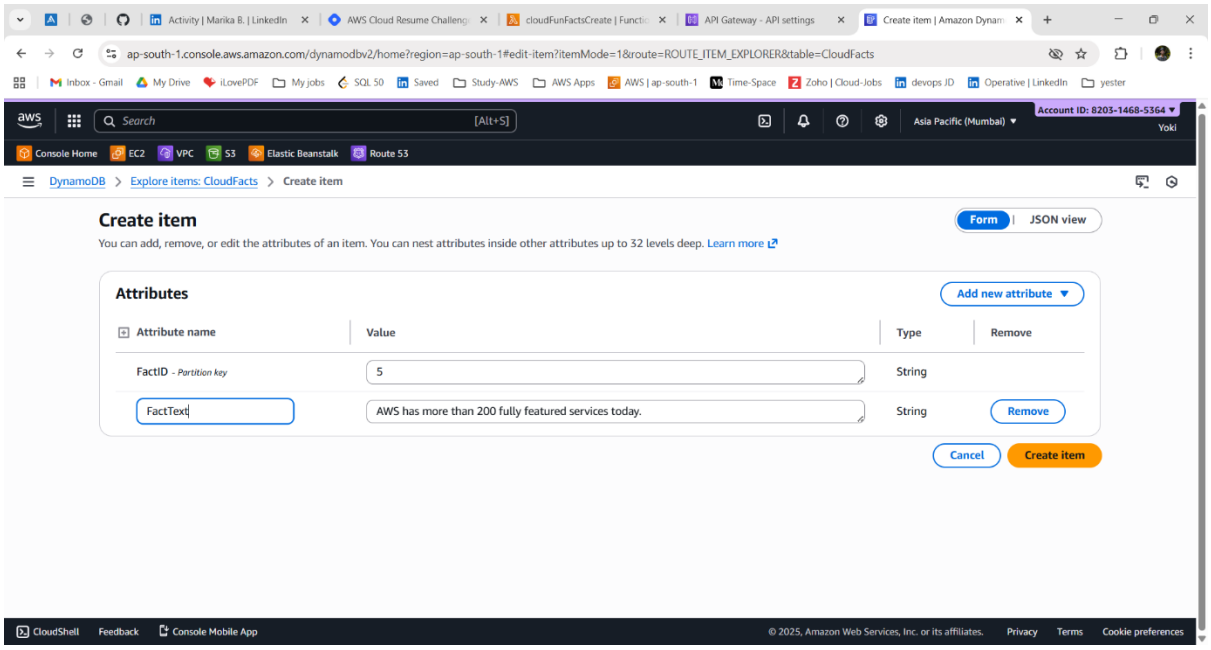
  FactText → the actual fun fact string.



Step - 2: Insert Sample Items:

~ Now, let's add a few sample fun facts to our database:

1. Go to your CloudFacts table → Explore table items → Create item.



2. Add entries like:

{ "FactID": "1", "FactText": "AWS S3 was one of the very first AWS services, launched in 2006." }

{ "FactID": "2", "FactText": "Netflix runs almost all of its infrastructure on AWS." }

{ "FactID": "3", "FactText": "Cloud computing can save companies up to 30% on IT costs." }

{ "FactID": "4", "FactText": "NASA uses AWS to store and share Mars mission data with the public." }

{ "FactID": "5", "FactText": "AWS has more than 200 fully featured services today." }

{ "FactID": "6", "FactText": "The 'cloud' doesn't float in the sky, it's made of giant data centers on Earth." }

{ "FactID": "7", "FactText": "More than 90% of Fortune 100 companies use AWS." }

{ "FactID": "8", "FactText": "AWS data centers are so secure that they require retina scans and 24/7 guards." }

{ "FactID": "9", "FactText": "Serverless doesn't mean there are no servers, it just means you don't manage them." }

{ "FactID": "10", "FactText": "Each AWS Availability Zone has at least one independent power source and cooling system." }

{ "FactID": "11", "FactText": "Amazon once accidentally took down parts of the internet when S3 had an outage in 2017." }

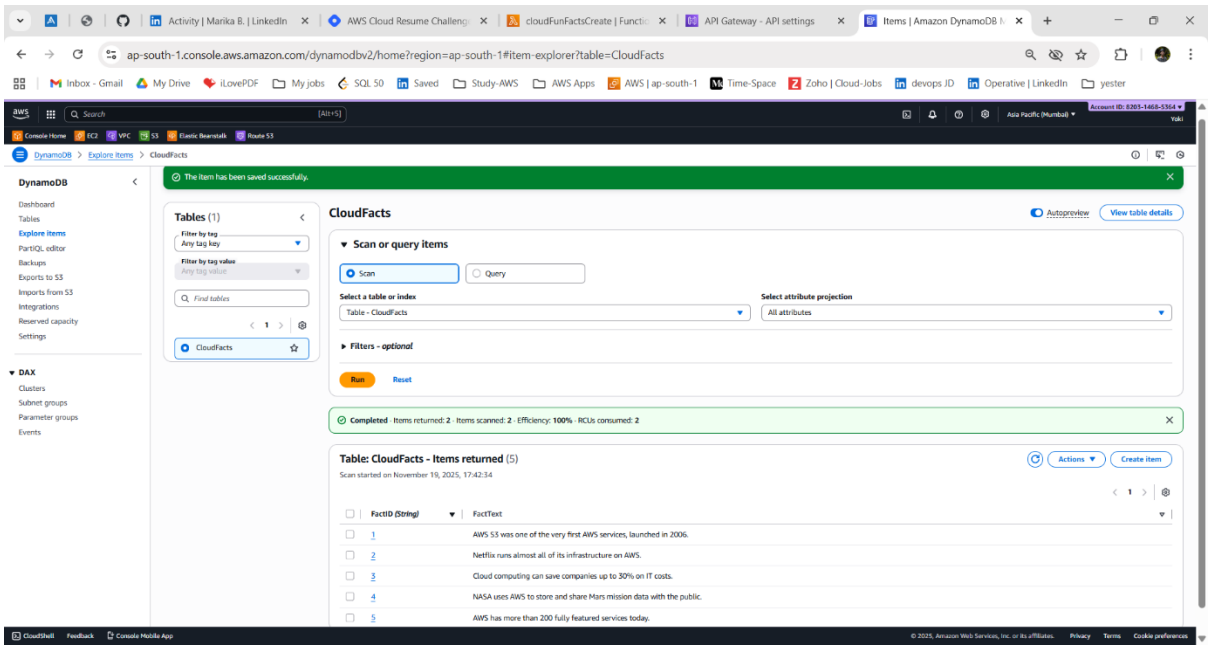
{ "FactID": "12", "FactText": "The fastest-growing AWS service is Amazon SageMaker, used for Machine Learning." }

{ "FactID": "13", "FactText": "The word 'cloud' became popular in the 1990s as a metaphor for the internet." }

{ "FactID": "14", "FactText": "AWS Lambda can automatically scale from zero to thousands of requests per second." }

{ "FactID": "15", "FactText": "More data is stored in the cloud today than in all personal computers combined."

}



👉 What's happening here?

- We're populating our database with facts.
- Later, instead of editing Lambda code, you can just add more rows here.

Step - 3: Update Lambda Role:

~ By default, our Lambda cannot talk to DynamoDB. We must grant it permission:

1. Go to **IAM → Roles**.

2. Find the role associated with your CloudFunFacts Lambda.

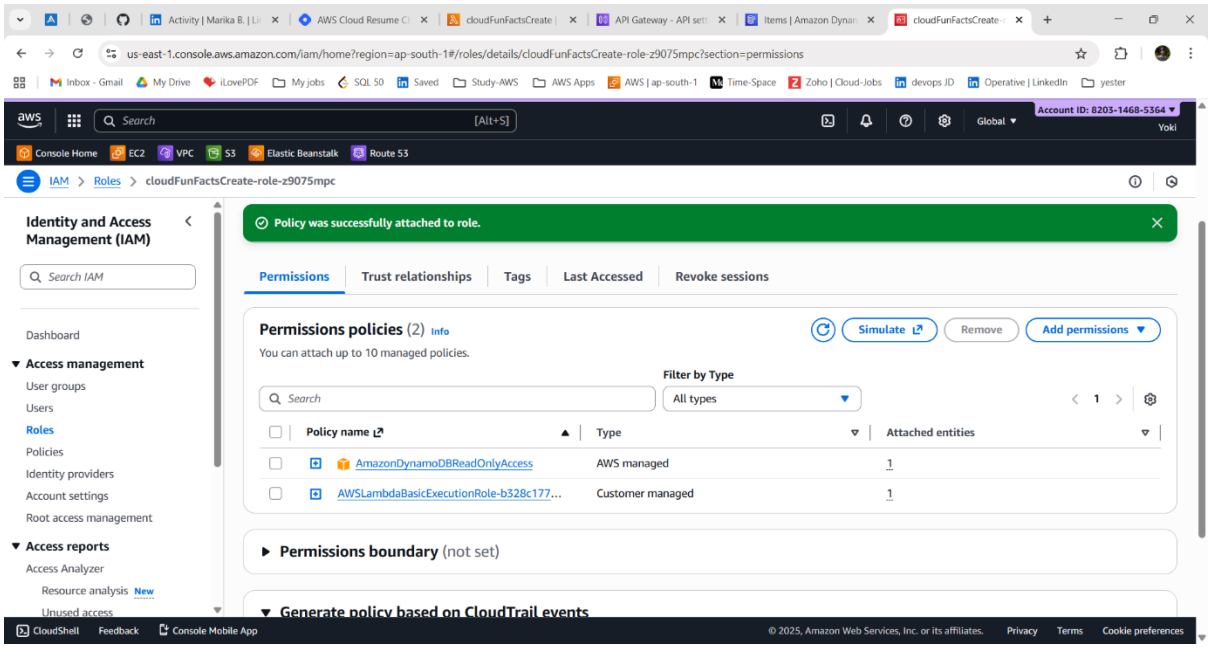
3. Attach policy: **AmazonDynamoDBReadOnlyAccess**.

→ You should see a random fun fact returned in the response.

👉 Why?

- AWS follows least privilege principle. Without explicit permissions, Lambda cannot access other AWS services.

- By attaching AmazonDynamoDBReadOnlyAccess, we're allowing Lambda to read items but not modify/delete them (safer).



👉 What's happening here?

- You're manually invoking the Lambda in the console

- AWS runs your code in a temporary environment, then destroys it (that's the beauty of serverless).

Step - 4: Update Lambda Code:

1. Replace your Lambda code with this:

Lambda Code:

```
 import boto3
import random
import json
# Create DynamoDB client
dynamodb = boto3.resource("dynamodb")
table = dynamodb.Table("CloudFacts")


def lambda_handler(event, context):
    # Scan entire table (not efficient for huge tables, but fine here)
    response = table.scan()
    items = response.get("Items", [])
    if not items:
        return {
            "statusCode": 500,
            "body": json.dumps({"error": "No facts found"})
        }
    # Pick random fact
    fact = random.choice(items)["FactText"]


    return {
        "statusCode": 200,
        "headers": {"Content-Type": "application/json"},
        "body": json.dumps({"fact": fact})
    }
```

2. Click **Deploy** to save the code.

👉 What's happening here?

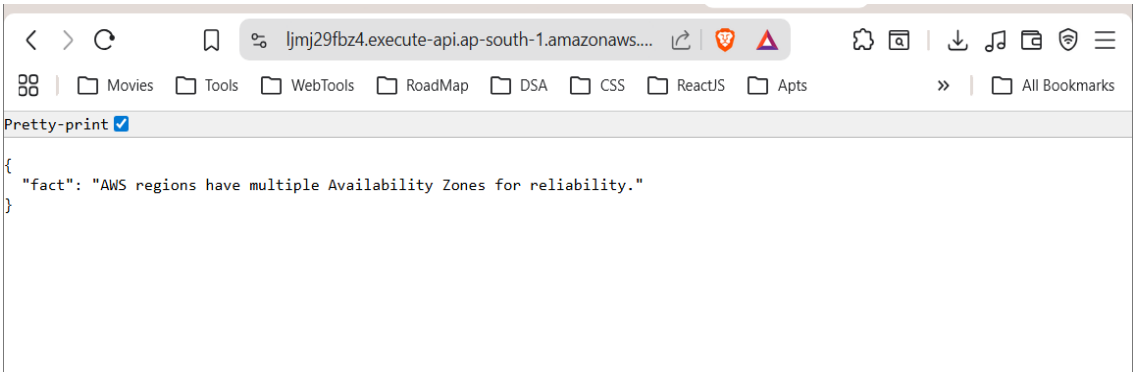
- boto3.resource("dynamodb") → Connects to DynamoDB.
- table.scan() → Reads all items from the table.
- random.choice(items) → Picks one random fact.
- Returns it as **JSON** response.

Step - 5: Test the API:

1. Go back to **API Gateway.**

2. Hit your endpoint:

Copy the URL (e.g., https://abcdedf.execute-api.ap-south -1.amazonaws.com).



This time, the response should come from DynamoDB. Example:

{

"fact": "NASA uses AWS to store and share Mars mission data with the public."

}

Outcome of Stage - 2:

→ At this point, your API is data-driven:

~ Fun facts are stored in DynamoDB.

~ Lambda fetches them dynamically.

~ You can add new facts without changing code.

~ This is exactly how modern apps evolve: starting with hardcoded logic, then moving to database-backed flexibility.