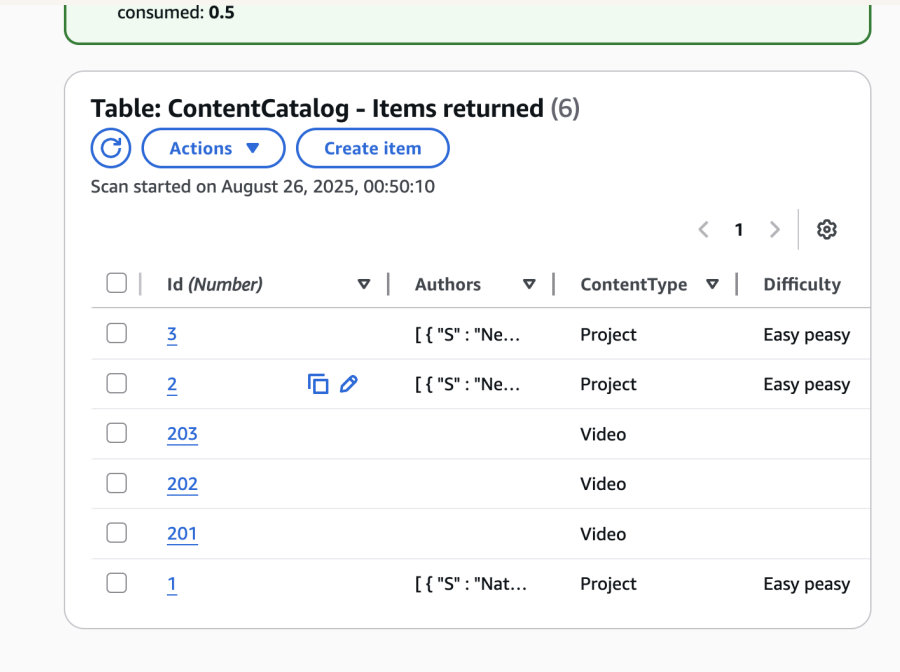
nextwork.org

Load Data into a DynamoDB Table

Kehinde Abiuwa



Kehinde Abiuwa
NextWork Student

Introducing Today's Project!

What is Amazon DynamoDB?

Amazon DynamoDB is a fully managed NoSQL database service provided by AWS that stores data in a key-value and document format. It is useful because it offers: High performance with single-digit millisecond response times. Flexibility, since items don’t require a fixed schema. Scalability, automatically handling large amounts of traffic and data. Fully managed operations, including backups, replication, and security without needing manual database administration. This makes it ideal for applications that need to scale quickly, handle unpredictable workloads, and maintain fast and reliable data access.

How I used Amazon DynamoDB in this project

ChatGPT said: In today's project, I used Amazon DynamoDB to create multiple tables, load sample data into them using AWS CloudShell, and explore how to view and edit items. This allowed me to practice working with a fully managed NoSQL database, understand how data is stored and retrieved, and see the flexibility of DynamoDB in handling different item attributes within the same table.

One thing I didn't expect in this project was...

One thing I didn't expect in this project is how DynamoDB tables can store items with completely different sets of attributes, unlike relational databases where every row must follow the same schema.

This project took me...

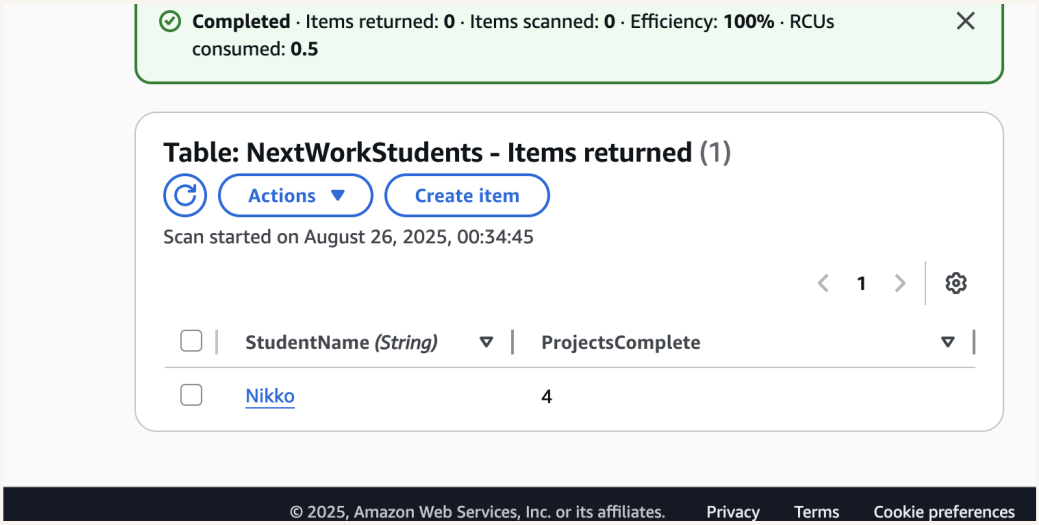
This project took me approximately 1 hour

K
Kehinde Abiuwa
NextWork Student

Create a DynamoDB table

DynamoDB tables organize data using items (rows) and attributes (columns), with each item uniquely identified by a primary key. The primary key can be: A partition key (hash key) – which determines the partition where the item is stored. Or a partition key + sort key (composite key) – which allows multiple items with the same partition key to be distinguished and sorted by the sort key. This structure makes DynamoDB highly scalable and efficient for fast lookups and queries.

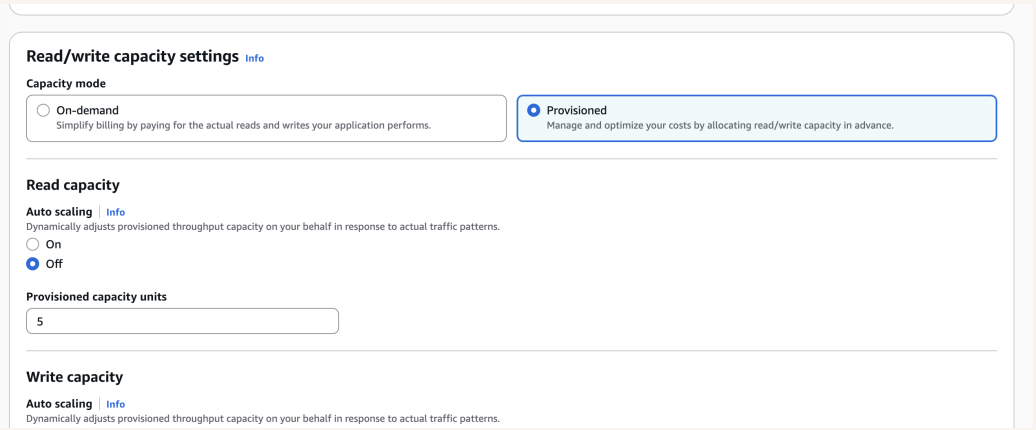
In DynamoDB, an attribute is like a piece of data about an item. In this case, our item is Nikko and the attribute is the number of projects Nikko completed.


Completed · Items returned: 0 · Items scanned: 0 · Efficiency: 100% · RCUs consumed: 0.5
Table: NextWorkStudents - Items returned (1)
Actions
Create item
Scan started on August 26, 2025, 00:34:45
1
StudentName (String)
ProjectsComplete
Nikko
4
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Read and Write Capacity

Read capacity units (RCUs) and write capacity units (WCUs) are the measures of throughput capacity in DynamoDB. Read Capacity Units (RCUs): One RCU represents one strongly consistent read per second, or two eventually consistent reads per second, for items up to 4 KB in size. Write Capacity Units (WCUs): One WCU represents one write per second (up to 1 KB of data) for items up to 1 KB in size. They are important because they define how many read and write operations your table can handle, directly affecting performance and cost

Amazon DynamoDB's Free Tier covers 25 GB of storage, along with up to 25 write capacity units (WCUs) and 25 read capacity units (RCUs) each month for free, for the first 12 months. I turned off auto scaling because I wanted to keep my read and write capacity fixed at a low, predictable level. This helps me stay within the Free Tier limits and avoid unexpected charges from DynamoDB automatically scaling capacity beyond what I need for this project.


Read/write capacity settings Info
Capacity mode
On-demand
Simplify billing by paying for the actual reads and writes your application performs.
Provisioned
Manage and optimize your costs by allocating read/write capacity in advance.
Read capacity
Auto scaling | Info
Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.
On
Off
Provisioned capacity units
5
Write capacity
Auto scaling | Info
Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.

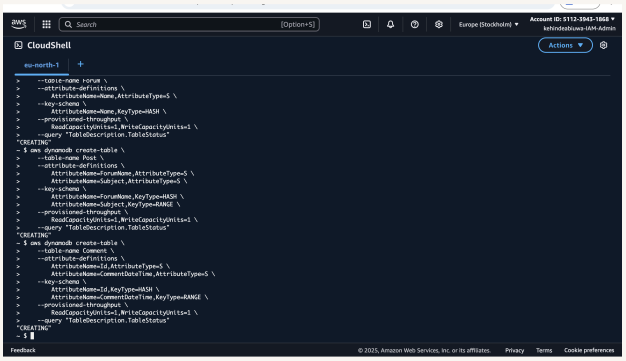
K
Kehinde Abiuwa
NextWork Student

Using CLI and CloudShell

AWS CloudShell is a browser-based command-line environment available directly in the AWS Management Console. It comes preconfigured with the AWS CLI, SDKs, and common developer tools, making it easy to run commands, manage resources, and automate tasks in AWS without needing to set up or configure anything on your local computer.

AWS CLI is the Amazon Web Services Command Line Interface, a unified tool that lets you interact with AWS services directly from a terminal or command prompt. It allows you to create, configure, and manage AWS resources by running commands, making it useful for automation, scripting, and managing infrastructure without relying solely on the AWS Management Console.

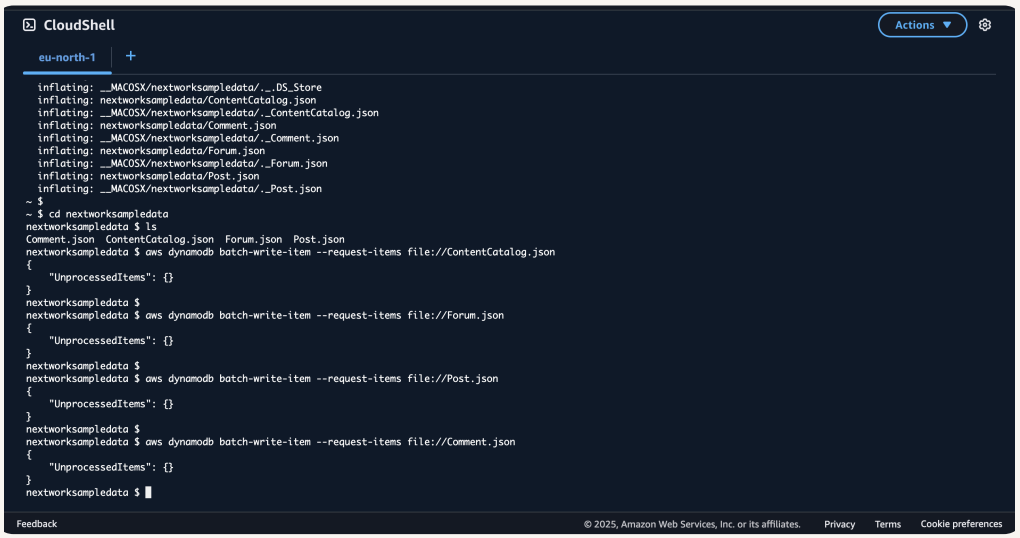
I ran a CLI command in AWS CloudShell that created four DynamoDB tables: ContentCatalog with Id as the partition key. Forum with Name as the partition key. Post with ForumName as the partition key and Subject as the sort key. Comment with Id as the partition key and CommentDateTime as the sort key. Each table was set up with provisioned throughput of 1 read capacity unit and 1 write capacity unit.


CloudShell
Actions
eu-north-1
--table-name Forum \
--attribute-definitions \
AttributeName=Name,AttributeType=S \
--key-schema \
AttributeName=Name,KeyType=HASH \
--provisioned-throughput \
ReadCapacityUnits=1,WriteCapacityUnits=1 \
--query "TableDescription.TableStatus"
"CREATING"
~ $ aws dynamodb create-table \
--table-name Post \
--attribute-definitions \
AttributeName=ForumName,AttributeType=S \
AttributeName=Subject,AttributeType=S \
--key-schema \
AttributeName=ForumName,KeyType=HASH \
AttributeName=Subject,KeyType=RANGE \
--provisioned-throughput \
ReadCapacityUnits=1,WriteCapacityUnits=1 \
--query "TableDescription.TableStatus"
"CREATING"
~ $ aws dynamodb create-table \
--table-name Comment \
--attribute-definitions \
AttributeName=Id,AttributeType=S \
AttributeName=CommentDateTime,AttributeType=S \
--key-schema \
AttributeName=Id,KeyType=HASH \
AttributeName=CommentDateTime,KeyType=RANGE \
--provisioned-throughput \
ReadCapacityUnits=1,WriteCapacityUnits=1 \
--query "TableDescription.TableStatus"
"CREATING"
~ $
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

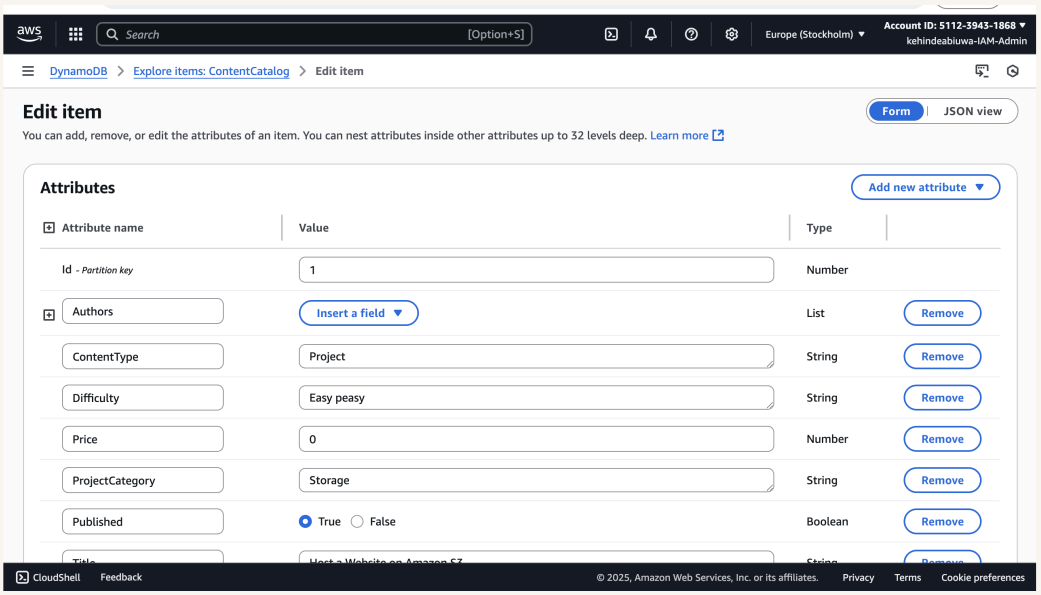
Kehinde Abiuwa
NextWork Student

Loading Data with CLI

I ran a CLI command in AWS CloudShell that first created DynamoDB tables (ContentCatalog, Forum, Post, and Comment), then downloaded and unzipped sample data files (JSON format), and finally used the batch-write-item command to load the sample data into each table. This allowed me to quickly populate my DynamoDB tables with test data for querying and updating.



Observing Item Attributes



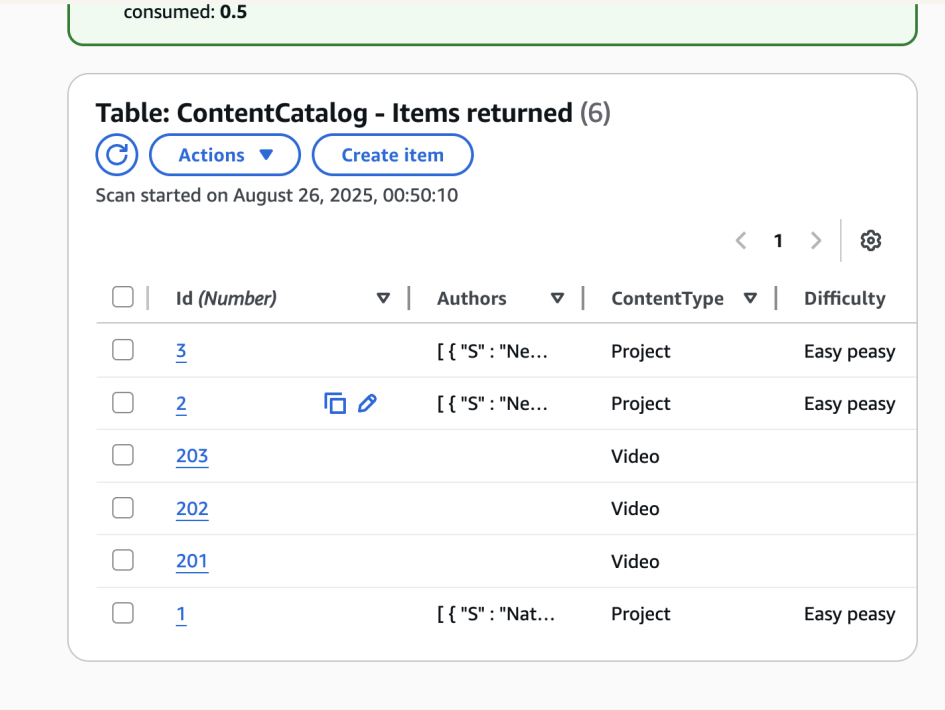
I checked a ContentCatalog item, which had the following attributes: Id (Partition key): 1 Authors: a list of authors ContentType: Project Difficulty: Easy peasy Price: 0 ProjectCategory: Storage Published: True Title: Host a Website on Amazon S3 These attributes describe a specific piece of content stored in the DynamoDB ContentCatalog table.

I checked another ContentCatalog item, which had a different set of attributes: Id (Partition key): 201 ContentType: Video Price: 0 Services: a list of services Title: AWS Relational vs Non Relational Databases in 6 minutes URL: https://youtu.be/SzvdEf9y0eA VideoType: Educational This shows how DynamoDB items in the same table don't need to have the exact same attributes, making it flexible for storing different types of content.

Benefits of DynamoDB

A benefit of DynamoDB over relational databases is flexibility, because items in the same table don't need to follow a fixed schema — each item can have different attributes. This makes it easy to adapt the database as application requirements change, without having to redesign tables or migrate data like you would in a relational database.

Another benefit over relational databases is speed, because DynamoDB uses a key-value access model with data distributed across partitions. Instead of performing complex joins or scans like relational databases, DynamoDB can retrieve items directly by their partition key, making lookups highly efficient and enabling single-digit millisecond response times at any scale.


consumed: 0.5
Table: ContentCatalog - Items returned (6)
Actions
Create item
Scan started on August 26, 2025, 00:50:10
1
Id (Number)
Authors
ContentType
Difficulty
3
[{ "S" : "Ne...
Project
Easy peasy
2
[{ "S" : "Ne...
Project
Easy peasy
203
Video
202
Video
201
Video
1
[{ "S" : "Nat...
Project
Easy peasy

The place to learn & showcase your skills

Check out nextwork.org for more projects