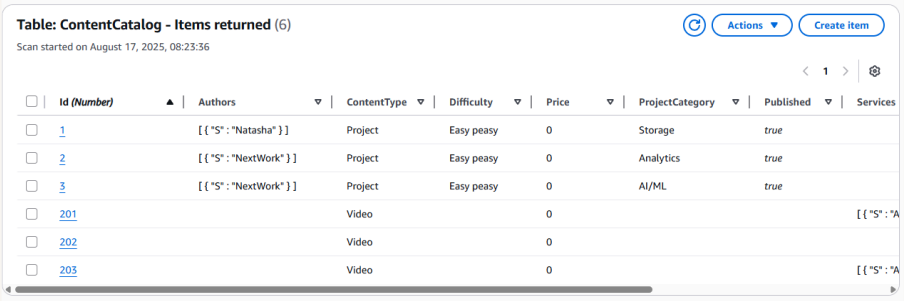
nextwork.org

Load Data into a DynamoDB Table

Louis Moyo



Louis Moyo
NextWork Student

Introducing Today's Project!

What is Amazon DynamoDB?

Amazon DynamoDB is a fully managed, serverless NoSQL database. It's useful because it's fast, scales automatically, and needs no servers or schema migrations.

How I used Amazon DynamoDB in this project

I created tables in CloudShell with the AWS CLI, bulk-loaded sample JSON using batch-write-item, then viewed and edited items in the console.

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

That items in the same table can have different attributes - no fixed columns - so I could add fields without changing a schema.

This project took me...

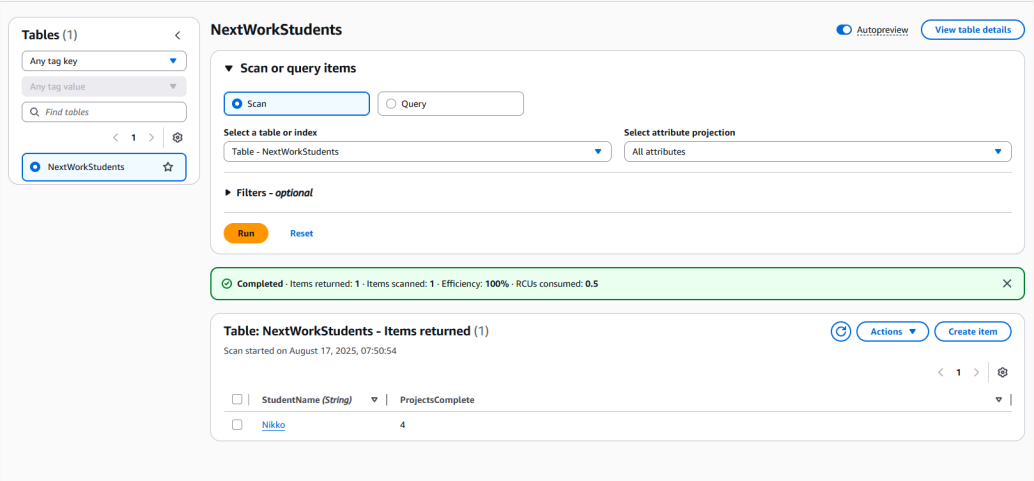
This project took me 1 hour.

Louis Moyo
NextWork Student

Create a DynamoDB table

DynamoDB organises data as a list of items (e.g., a student like “Nikko”), each with its own attributes (e.g., ProjectsComplete). Items don’t need the same attributes so, it’s not fixed rows/columns like a relational DB.

An attribute is one piece of info about an item (a field and its value). Example: StudentName = "Nikko" or ProjectsComplete = 4.

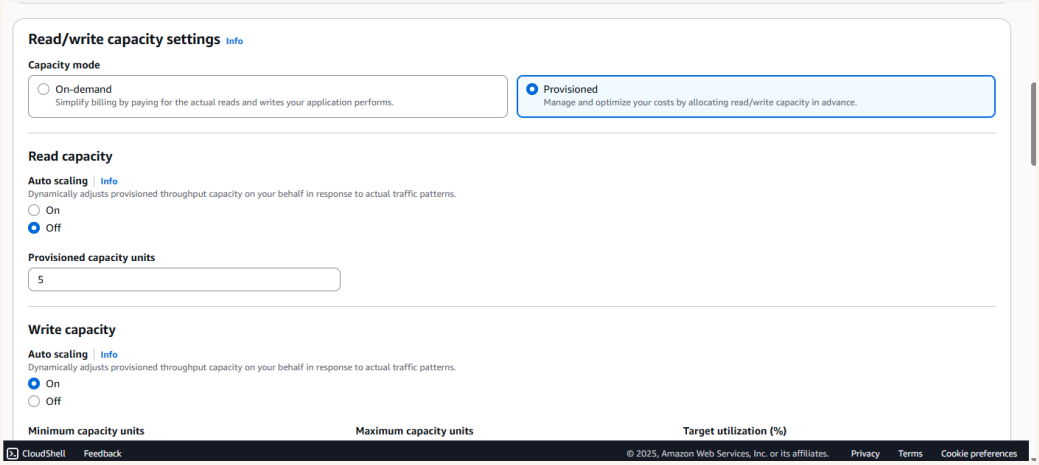


Louis Moyo
NextWork Student

Read and Write Capacity

Read capacity units (RCUs) and write capacity units (WCUs) are how DynamoDB measures speed you reserve. RCU = read speed. Roughly 2 reads/second for small items. WCU = write speed. About 1 write/second for small items.

The Free Tier gives you 25 GB storage plus 25 RCUs and 25 WCUs each month (enough for small demos). I turned auto scaling off so the table won't scale up beyond these limits and risk charges, fixed 1 RCU / 1 WCU is plenty for this project.



Louis Moyo
NextWork Student

Using CLI and CloudShell

AWS CloudShell is a browser shell in AWS for running CLI commands - no setup.

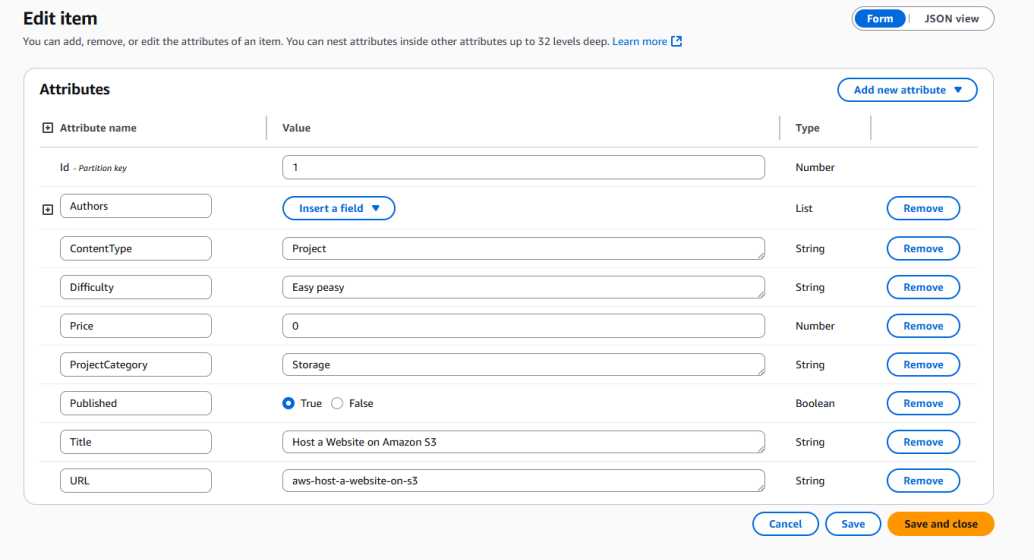
AWS CLI is a command-line tool to manage AWS resources from your terminal.

I used AWS CloudShell to run AWS CLI commands that created a handful of DynamoDB tables for our app. The commands told AWS to make empty tables, give each one a basic primary key, and set small capacity so they're cheap/free.

Loading Data with CLI

I ran a CLI command in AWS CloudShell that bulk-loaded data into DynamoDB using aws dynamodb batch-write-item from JSON files.

Observing Item Attributes



I checked a ContentCatalog item, which had the following attributes: Id (Number, partition key) Authors (List) ContentType (String) Difficulty (String) Price (Number) ProjectCategory (String) Published (Boolean) Title (String) URL (String)

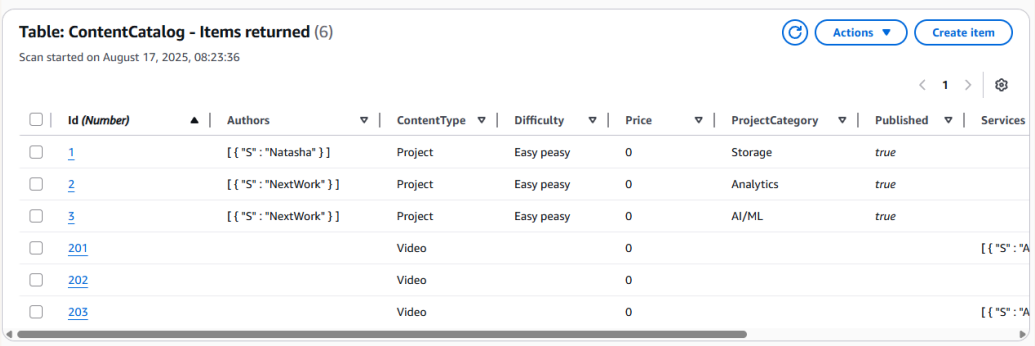
I checked another ContentCatalog item, which had a different set of attributes: Id (Number, partition key), ContentType = Video (String), Price (Number), Services (List), Title (String), URL (String), and VideoType (String). This shows items can vary.

Louis Moyo
NextWork Student

Benefits of DynamoDB

A benefit is flexibility, because DynamoDB is schemaless - each item can have different attributes, and you can add new fields anytime without migrations.

It's fast because it does key-based lookups and partitions data across many servers, avoiding joins and scaling horizontally for single-digit millisecond reads/writes.



The place to learn & showcase your skills

Check out nextwork.org for more projects