



# Load Data into a DynamoDB Table



Louis Moyo

Table: ContentCatalog - Items returned (6)							
Scan started on August 17, 2025, 08:23:36							
<input type="checkbox"/>	<b>Id (Number)</b>	Authors	Content Type	Difficulty	Price	Project Category	Published
<input type="checkbox"/>	<a href="#">1</a>	[{"S": "Natasha"}]	Project	Easy peasy	0	Storage	true
<input type="checkbox"/>	<a href="#">2</a>	[{"S": "NextWork"}]	Project	Easy peasy	0	Analytics	true
<input type="checkbox"/>	<a href="#">3</a>	[{"S": "NextWork"}]	Project	Easy peasy	0	AI/ML	true
<input type="checkbox"/>	<a href="#">201</a>		Video		0		[{"S": "A"}]
<input type="checkbox"/>	<a href="#">202</a>		Video		0		[{"S": "A"}]
<input type="checkbox"/>	<a href="#">203</a>		Video		0		[{"S": "A"}]



# 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.



# 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.

The screenshot shows the AWS DynamoDB console interface. On the left, a sidebar titled 'Tables (1)' lists a single table named 'NextWorkStudents'. The main area is titled 'NextWorkStudents' and contains a 'Scan or query items' section. Under this section, the 'Scan' button is selected. Below the buttons are dropdown menus for 'Select a table or index' (set to 'Table - NextWorkStudents') and 'Select attribute projection' (set to 'All attributes'). At the bottom of this section are 'Run' and 'Reset' buttons. A green status bar at the bottom indicates: 'Completed - Items returned: 1 - Items scanned: 1 - Efficiency: 100% - RCU consumed: 0.5'. Below this, a table titled 'Table: NextWorkStudents - Items returned (1)' shows one item: 'Nikko' with 'ProjectsComplete' set to '4'. There are 'Actions' and 'Create item' buttons above the table, and navigation controls (back, forward, search) below it.



# 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.

**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**

**Write capacity**

**Auto scaling** Info  
Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.

On

Off

Minimum capacity units	Maximum capacity units	Target utilization (%)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

# 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.

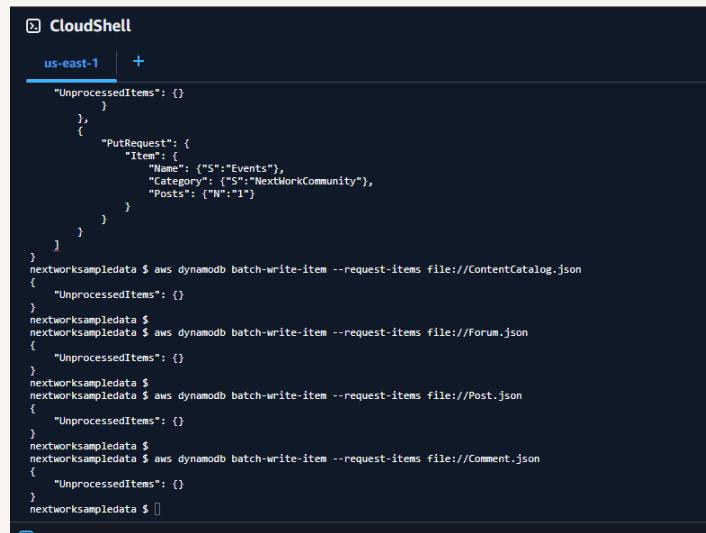
I used AWS CloudShell to run AWS CLI commands that created a handful of DynamoDB tables for our app. The commands told AWS to make empty tables, give each one a basic primary key, and set small capacity so they're cheap/free.



```
aws dynamodb create-table \
    --table-name ContentRating \
    --attribute-definitions AttributeDefinition \
    --key-schema KeySchema \
    --provisioned-throughput ProvisionedThroughput \
    query "tableDescription.tableStatus" \
CREATE
aws dynamodb create-table \
    --table-name Forum \
    --attribute-definitions AttributeDefinition \
    --key-schema KeySchema \
    --provisioned-throughput ProvisionedThroughput \
    query "tableDescription.tableStatus" \
CREATE
aws dynamodb create-table \
    --attribute-definitions AttributeDefinition \
    --key-schema KeySchema \
    --provisioned-throughput ProvisionedThroughput \
    query "tableDescription.tableStatus" \
CREATE
aws dynamodb create-table \
    --table-name ForumPost \
    --attribute-definitions AttributeDefinition \
    --key-schema KeySchema \
    --provisioned-throughput ProvisionedThroughput \
    query "tableDescription.tableStatus" \
CREATE
aws dynamodb create-table \
    --table-name Post \
    --attribute-definitions AttributeDefinition \
    --key-schema KeySchema \
    --provisioned-throughput ProvisionedThroughput \
    query "tableDescription.tableStatus" \
CREATE
```

# 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.



```
CloudShell
us-east-1 | +
"UnprocessedItems": {}
},
{
    "PutRequest": {
        "Item": {
            "Name": {"S":"Events"},
            "Category": {"S":"NextWorkCommunity"},
            "Posts": {"N":"1"}
        }
    }
}
nextworksampled $ aws dynamodb batch-write-item --request-items file://ContentCatalog.json
{
    "UnprocessedItems": {}
}
nextworksampled $
nextworksampled $ aws dynamodb batch-write-item --request-items file://Forum.json
{
    "UnprocessedItems": {}
}
nextworksampled $
nextworksampled $ aws dynamodb batch-write-item --request-items file://Post.json
{
    "UnprocessedItems": {}
}
nextworksampled $
nextworksampled $ aws dynamodb batch-write-item --request-items file://Comment.json
{
    "UnprocessedItems": {}
}
nextworksampled $ []
```



# Observing Item Attributes

**Edit item**  
You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. [Learn more](#)

[Form](#) | [JSON view](#)

Attribute name	Value	Type	Add new attribute ▾
Id - Partition key	1	Number	<a href="#">Remove</a>
Authors	<a href="#">Insert a field ▾</a>	List	<a href="#">Remove</a>
ContentType	Project	String	<a href="#">Remove</a>
Difficulty	Easy peasy	String	<a href="#">Remove</a>
Price	0	Number	<a href="#">Remove</a>
ProjectCategory	Storage	String	<a href="#">Remove</a>
Published	<input checked="" type="radio"/> True <input type="radio"/> False	Boolean	<a href="#">Remove</a>
Title	Host a Website on Amazon S3	String	<a href="#">Remove</a>
URL	aws-host-a-website-on-s3	String	<a href="#">Remove</a>

[Cancel](#) [Save](#) [Save and close](#)

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.

# 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.

Table: ContentCatalog - Items returned (6)						
Scan started on August 17, 2025, 08:23:36						
<input type="checkbox"/>	Id (Number)	Authors	ContentType	Difficulty	Price	ProjectCategory
<input type="checkbox"/>	<a href="#">1</a>	[{"S": "Natasha"}]	Project	Easy peasy	0	Storage
<input type="checkbox"/>	<a href="#">2</a>	[{"S": "NextWork"}]	Project	Easy peasy	0	Analytics
<input type="checkbox"/>	<a href="#">3</a>	[{"S": "NextWork"}]	Project	Easy peasy	0	AI/ML
<input type="checkbox"/>	<a href="#">201</a>		Video		0	[{"S": "A"}]
<input type="checkbox"/>	<a href="#">202</a>		Video		0	[{"S": "A"}]
<input type="checkbox"/>	<a href="#">203</a>		Video		0	[{"S": "A"}]



[nextwork.org](https://nextwork.org)

# The place to learn & showcase your skills

Check out [nextwork.org](https://nextwork.org) for more projects

