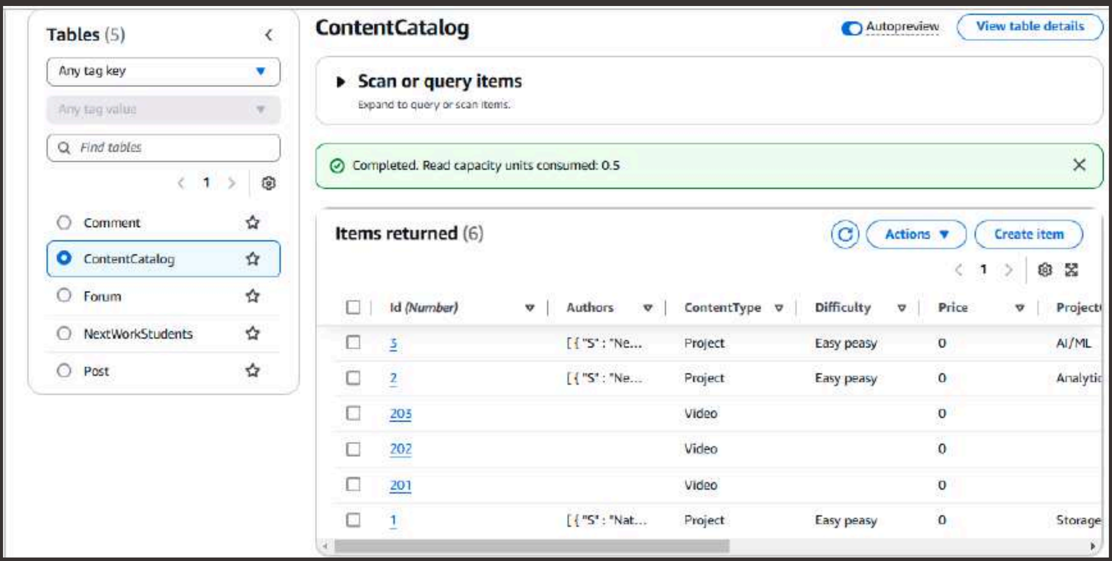
nextwork.org

Load Data into a DynamoDB Table

Nicolás Aversa



ContentCatalog

Scan or query items
Expand to query or scan items.

Completed. Read capacity units consumed: 0.5

Items returned (6)

Id (Number)	Authors	ContentType	Difficulty	Price	Project
3	[{ "S" : "Ne...	Project	Easy peasy	0	AI/ML
2	[{ "S" : "Ne...	Project	Easy peasy	0	Analytic
203		Video		0	
202		Video		0	
201		Video		0	
1	[{ "S" : "Nat...	Project	Easy peasy	0	Storage

Introducing Today's Project!

What is Amazon DynamoDB?

Amazon DynamoDB is a non-relational database service that lets you store and retrieve large amounts of data quickly, scale seamlessly to handle high traffic, and use a flexible schema where each item can have different attributes.

How I used Amazon DynamoDB in this project

I used Amazon DynamoDB in today's project to create tables, load them with items and its attributes using AWS CloudShell and then making some manual updates to the attributes.

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

One thing I didn't expect in this project was to realize how intuitive and user-friendly Amazon DynamoDB is. It was really easy to use!

This project took me...

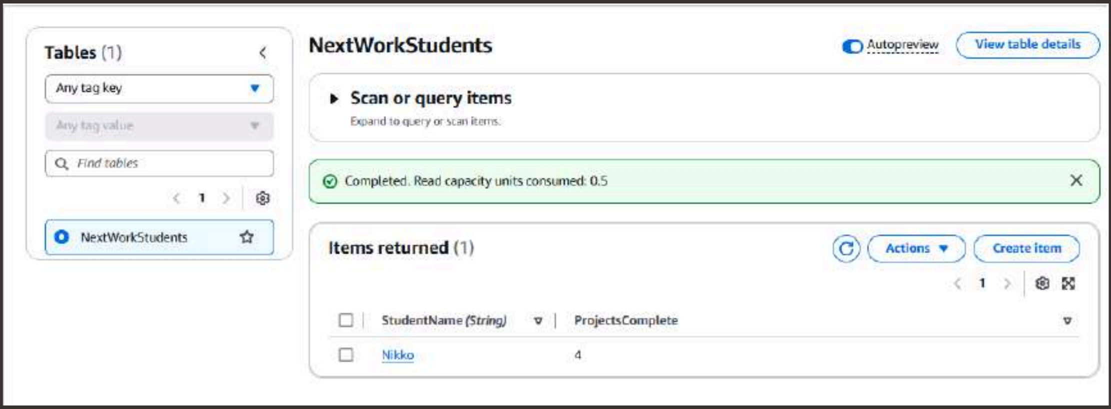
This project took me an hour.

Nicolás Aversa
NextWork Student

Create a DynamoDB table

DynamoDB tables organises data using items and attributes. It consists of a list of items where each item has their own list of attributes. It's super flexible.

An attribute is a piece of data about an item. It is like a property of the item, and it can have different types of values.



NextWorkStudents

Completed. Read capacity units consumed: 0.5

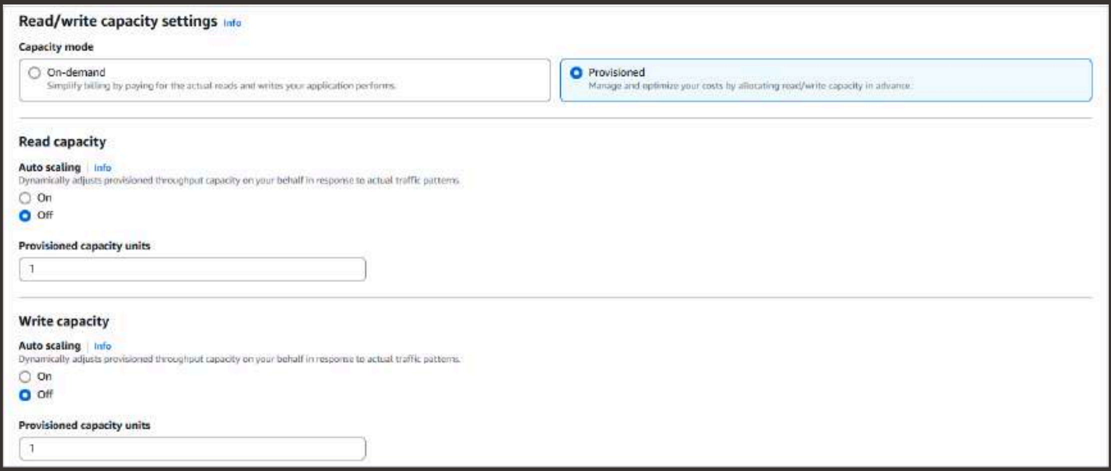
Items returned (1)

StudentName (String)	ProjectsComplete
Nikko	4

Read and Write Capacity

Read capacity units (RCUs) and write capacity units (WCUs) are the engines DynamoDB uses to handle your reads/write needs. Depending on how many Item read/second or write/second you select, they vary.

Amazon DynamoDB's Free Tier covers up to 25GB of data storage, plus 25 Write and 25 Read Capacity Units (WCU, RCU). I turned off auto scaling because if you don't monitor that feature, it can make your throughput capacity go beyond AWS Free Tier.



Read/write capacity settings Info

Capacity mode

On-demand [✓ Provisioned]

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

1

Write capacity

Auto scaling | Info

Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.

On [✓ Off]

Provisioned capacity units

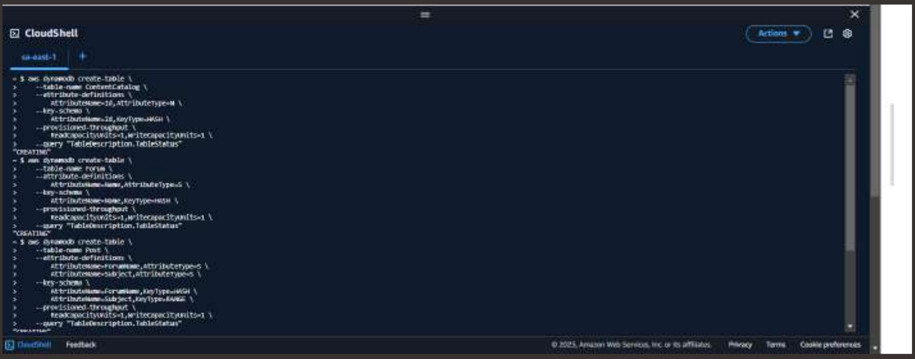
1

Using CLI and CloudShell

AWS CloudShell is a shell that provides a way of interacting with AWS Management Console through the use of commands. It provides more speed and versatility, since you can use scripts instead of interacting manually with AWS resources.

AWS CLI is a software that lets you create, delete and update AWS resources with commands instead of clicking through your console.

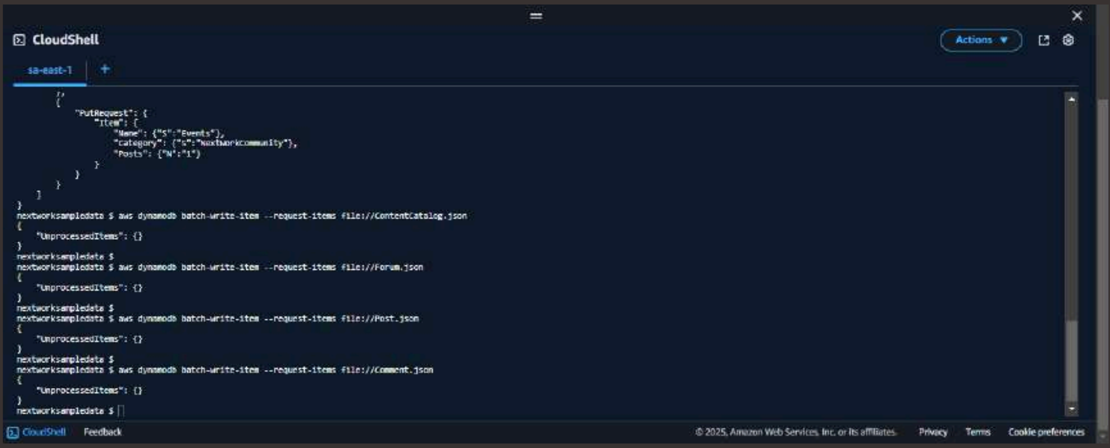
I ran a CLI command in AWS CloudShell that created four new tables in AWS DynamoDB, each with specific attributes and settings. Those are ContentCatalog, Forum, Post, and Comment tables.



```
~ $ aws dynamodb create-table \
>     --table-name ContentCatalog \
>     --attribute-definitions \
>         AttributeName=Id,AttributeType=N \
>     --key-schema \
>         AttributeName=Id,KeyType=HASH \
>     --provisioned-throughput \
>         ReadCapacityUnits=1,WriteCapacityUnits=1 \
>     --query "TableDescription.TableStatus"
"CREATING"
~ $ aws dynamodb create-table \
>     --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"
```

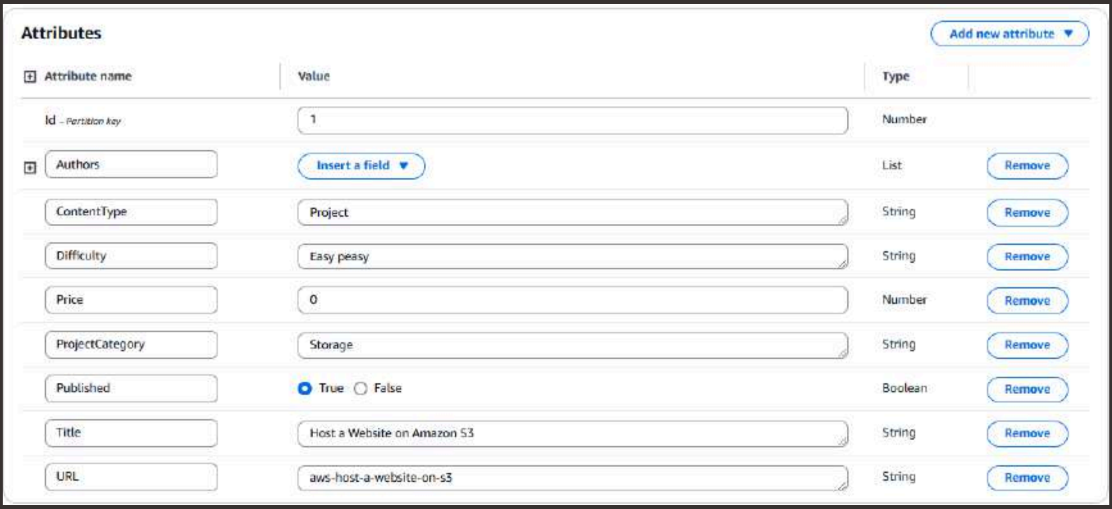
Loading Data with CLI

I ran a CLI command in AWS CloudShell that loads the data of all 4 files into DynamoDB. 'aws dynamodb batch-write-item --request-items file://FILENAME.json' means "upload multiple items from a local JSON file into their specified tables in one go."



```
        },
        {
            "PutRequest": {
                "Item": {
                    "Name": {"S":"Events"},
                    "category": {"S":"NextworkCommunity"},
                    "Posts": {"N":"1"}
                }
            }
        }
    ]
}
nextworksampledata $ aws dynamodb batch-write-item --request-items file://ContentCatalog.json
{
    "UnprocessedItems": {}
}
nextworksampledata $
nextworksampledata $ aws dynamodb batch-write-item --request-items file://Forum.json
{
    "UnprocessedItems": {}
}
nextworksampledata $
nextworksampledata $ aws dynamodb batch-write-item --request-items file://Post.json
{
    "UnprocessedItems": {}
}
nextworksampledata $
nextworksampledata $ aws dynamodb batch-write-item --request-items file://Comment.json
{
    "UnprocessedItems": {}
}
nextworksampledata $
```

Observing Item Attributes



Attributes

Add new attribute ▼

Attribute name	Value	Type	
Id - Partition key	1	Number	
Authors	Insert a field ▼	List	Remove
ContentType	Project	String	Remove
Difficulty	Easy peasy	String	Remove
Price	0	Number	Remove
ProjectCategory	Storage	String	Remove
Published	[✓ True] False	Boolean	Remove
Title	Host a Website on Amazon S3	String	Remove
URL	aws-host-a-website-on-s3	String	Remove

I checked a ContentCatalog item, which had the following attributes: 'Id', 'Authors', 'ContentType', 'Difficulty', 'Price', 'ProjectCategory', 'Published', 'Title', and 'URL'.

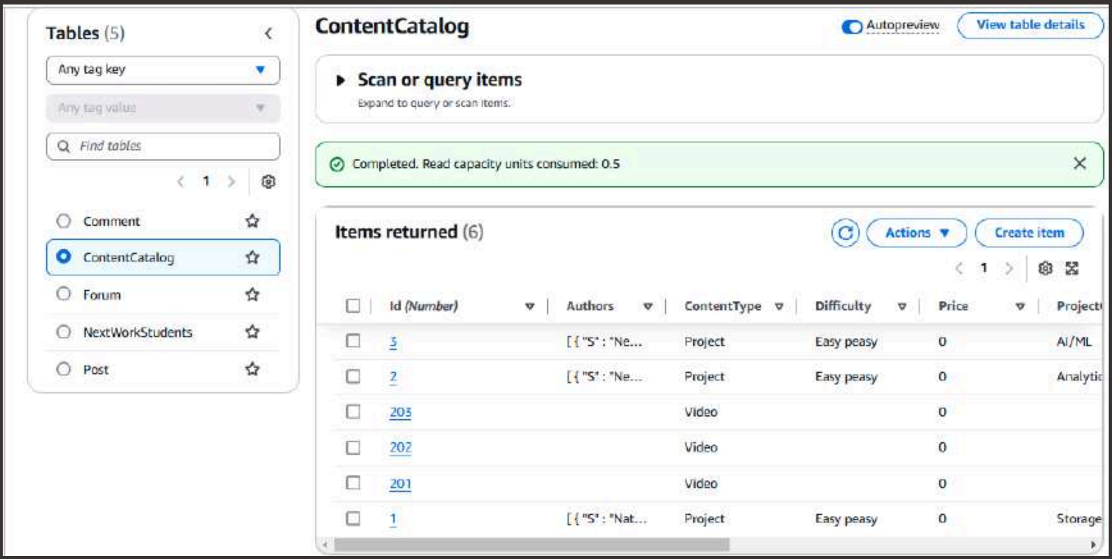
I checked another ContentCatalog item, which had a different set of attributes: 'Id', 'ContentType', 'Price', 'Services', 'Title', 'URL', and 'VideoType'.

Nicolás Aversa
NextWork Student

Benefits of DynamoDB

A benefit of DynamoDB over relational databases is flexibility, because every item having their own unique set of attributes is a huge advantage when items in a table could look different from each other.

Another benefit over relational databases is speed, because tables can use partition keys to split up a table and quickly find the items they're looking for. Relational databases have to scan through the entire table to find data, making it slower.



ContentCatalog

Scan or query items
Expand to query or scan items.

Completed. Read capacity units consumed: 0.5

Items returned (6)

Id (Number)	Authors	ContentType	Difficulty	Price	Project
3	[{"S": "Ne...	Project	Easy peasy	0	AI/ML
2	[{"S": "Ne...	Project	Easy peasy	0	Analytic
203		Video		0	
202		Video		0	
201		Video		0	
1	[{"S": "Nat...	Project	Easy peasy	0	Storage

Everyone should be in a job they love.

Check out nextwork.org for more projects