



Load Data into DynamoDB



nikhil7_94@hotmail.com

| Items returned (6) | | | | |
|--------------------|----------------------------------|--|--|--------------------|
| | <input type="button" value="C"/> | <input type="button" value="Actions ▾"/> | <input type="button" value="Create item"/> | |
| | <input type="checkbox"/> | Id (Number) | Authors | ContentType |
| | <input type="checkbox"/> | <u>3</u> | [{"S": "NextWork"}] | Project |
| | <input type="checkbox"/> | <u>2</u> | [{"S": "NextWork"}] | Project |
| | <input type="checkbox"/> | <u>203</u> | | Video |
| | <input type="checkbox"/> | <u>202</u> | | Video |
| | <input type="checkbox"/> | <u>201</u> | | Video |
| | <input type="checkbox"/> | <u>1</u> | [{"S": "Natasha"}] | Project |

Introducing Today's Project!

What is Amazon DynamoDB?

DynamoDB is described as a NoSQL database or a key-value database. A NoSQL database is useful because it means you would not use SQL to query it while key-value is a specific way to store data that's flexible and efficient.

How I used Amazon DynamoDB in this project

Today's project I used Amazon DynamoDB to create tables and its items inside. Also I learned how to add attributes to the items though the AWS console. All this can be done using the AWS cloudshell as well with commands.

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

One thing I didnt expect was that tables with items and attributes can be added to AWS Dynamo DB using the cloudshell.

This project took me...

I took 2 hours with this project.

Create a DynamoDB table

DynamoDB tables organises data using primary keys which consist of partition keys and optional sort keys allowing for efficient data retrieval and scalability.

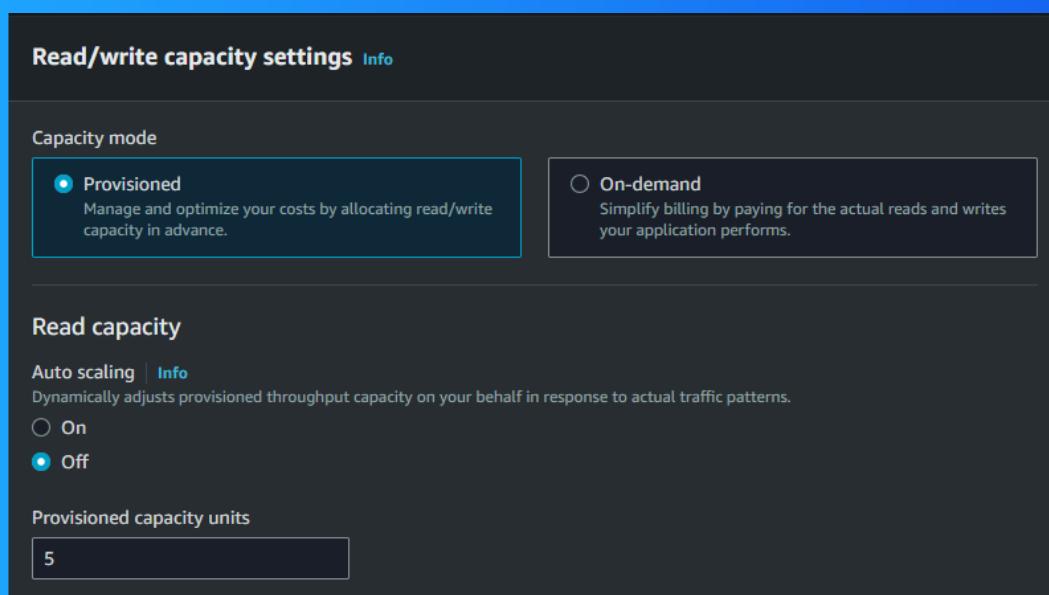
An attribute is like a piece of data about an item. In this case our item is Nikko and the attribute is the number of projects Nikko completed.

| Items returned (1) | | C | Actions ▾ | Create item |
|--|-------|---|-----------|-------------|
| < | 1 | > | | ⚙️ ✎ |
| <input type="checkbox"/> StudentName (String) ▾ ProjectsComplete ▾ | Nikko | 4 | | |

Read and Write Capacity

Read capacity units (RCUs) are a measure of how many engines DynamoDB is using to operate. Write capacity units (WCUs) are just like read capacity units but they give your DynamoDB tables the engines to edit/update/delete data.

Amazon DynamoDB's Free Tier covers up to 200M requests. I turned off auto scaling because I dont want to adjust dynamically the capacity throughput behalf in response to actual traffic patterns.

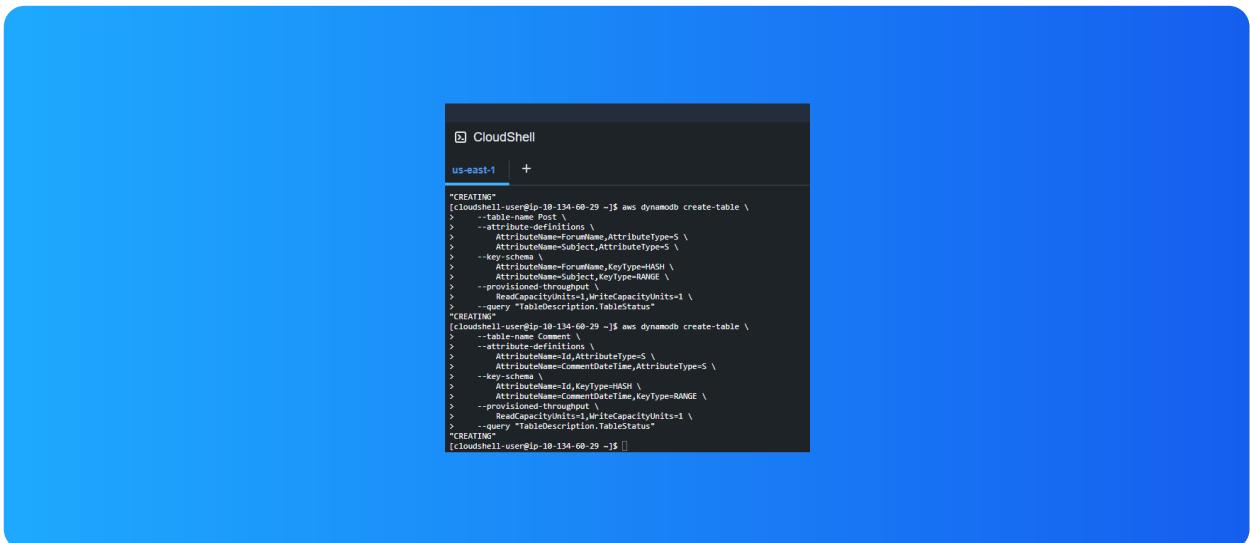


Using CLI and CloudShell

AWS CloudShell is shell in your AWS Management Console which means it's a space for you to run code. AWS CloudShell already has AWS CLI pre-installed.

AWS CLI is 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 new tables.



```
"CREATING"
[cloudshell-user@ip-10-134-60-29 ~]$ aws dynamodb create-table \
>   --table-name Post \
>   --attribute-definitions \
>     AttributeName=PostName,AttributeType=S \
>     AttributeName=PostSubject,AttributeType=S \
>   --key-schema \
>     AttributeName=PostName,KeyType=HASH \
>     AttributeName=PostSubject,KeyType=RANGE \
>   --provisioned-throughput \
>     ReadCapacityUnits=1,WriteCapacityUnits=1 \
>   --query "TableDescription.TableStatus"
"CREATING"
[cloudshell-user@ip-10-134-60-29 ~]$ aws dynamodb create-table \
>   --table-name Comment \
>   --attribute-definitions \
>     AttributeName=CommentId,AttributeType=S \
>     AttributeName=CommentDeleteTime,AttributeType=S \
>   --key-schema \
>     AttributeName=CommentId,KeyType=HASH \
>     AttributeName=CommentDeleteTime,KeyType=RANGE \
>   --provisioned-throughput \
>     ReadCapacityUnits=1,WriteCapacityUnits=1 \
>   --query "TableDescription.TableStatus"
"CREATING"
[cloudshell-user@ip-10-134-60-29 ~]$ [
```

Loading Data with CLI

I ran a CLI command in AWS CloudShell that download and unzip a zip file with data. Then I load the data of all four files into DynamoDB using AWS CLI's batch-write-item command.

```
[cloudshell-user@ip-10-134-60-29 nextworksampleddata]$ aws dynamodb batch-write-item --request-items file://ContentCatalog.json
{
    "UnprocessedItems": {}
}
[cloudshell-user@ip-10-134-60-29 nextworksampleddata]$ aws dynamodb batch-write-item --request-items file://Forum.json
{
    "UnprocessedItems": {}
}
[cloudshell-user@ip-10-134-60-29 nextworksampleddata]$ aws dynamodb batch-write-item --request-items file://Post.json
{
    "UnprocessedItems": {}
}
[cloudshell-user@ip-10-134-60-29 nextworksampleddata]$ aws dynamodb batch-write-item --request-items file://Comment.json
{
    "UnprocessedItems": {}
}
[cloudshell-user@ip-10-134-60-29 nextworksampleddata]$ █
```

Observing Item Attributes

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

Cancel Save Save and close

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

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

Benefits of DynamoDB

A benefit of DynamoDB over relational databases is flexibility, because it allows for schema-less data models making it easy to store and manage unstructured or semi-structured data.

Another benefit over relational databases is speed, because DynamoDB 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.

| Items returned (6) | | | | |
|--------------------------|-------------|---------------------|-------------|------------|
| | C | Actions ▾ | Create item | |
| | < | 1 | > | ⚙️ 🔍 |
| | Id (Number) | Authors | ContentType | Difficulty |
| <input type="checkbox"/> | <u>3</u> | [{"S": "NextWork"}] | Project | Easy peasy |
| <input type="checkbox"/> | <u>2</u> | [{"S": "NextWork"}] | Project | Easy peasy |
| <input type="checkbox"/> | <u>203</u> | | Video | |
| <input type="checkbox"/> | <u>202</u> | | Video | |
| <input type="checkbox"/> | <u>201</u> | | Video | |
| <input type="checkbox"/> | <u>1</u> | [{"S": "Natasha"}] | Project | Easy peasy |



NextWork.org

Everyone should be in a job they love.

Check out nextwork.org for
more projects

