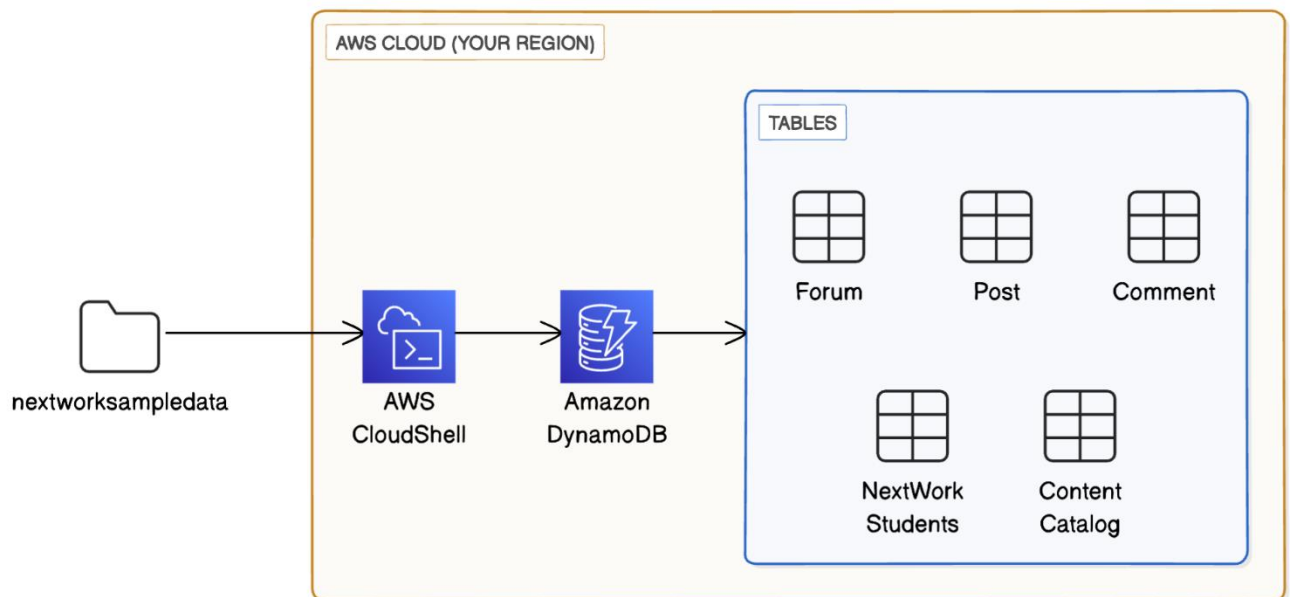


[NextWork.org](https://www.nextwork.org)

Load Data into DynamoDB

**Prajit Venkatachalam**<https://www.linkedin.com/in/prajit-venkatachalam/>



Introducing Today's Project!

What is Amazon DynamoDB?

Amazon DynamoDB is a non-relational database service provided by AWS that stores data in key-value pairs for efficient organization and retrieval.

How I used Amazon DynamoDB in this project

In today's project, I used DynamoDB to create five tables. I utilized CloudShell and the AWS CLI to quickly load data into the tables. I also compared DynamoDB to a relational database by exploring how items and attributes function differently.

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

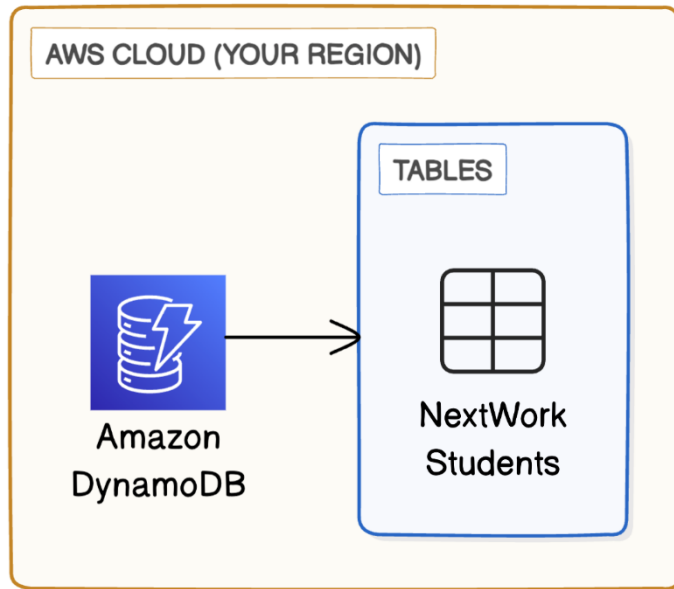
One thing I noted in this project was that I could perform many actions using the AWS CLI instead of the console. For example, I created tables, loaded data, and deleted tables all through the CLI.

This project took me...

This project took me 2.5 hours to complete including documentation.



Create a DynamoDB table



DynamoDB tables organize data using items and attributes. Each item is recorded with a set of attributes. Items can have any number of attributes, with a minimum of one (the partition key value).

Attributes are a piece of data about an item in a DynamoDB table. For example, if an item represents a specific student, the attributes could be StudentName, ProjectsCompleted, Email etc. The number of attributes for each item is flexible.

✔ Completed. Read capacity units consumed: 0.5

Items returned (1)

⌂ Actions Create item

< 1 > ⚙️

<input type="checkbox"/>	StudentName (String)	ProjectsComplete
<input type="checkbox"/>	Prajit	20



Read and Write Capacity

Read Capacity Units (RCUs) and Write Capacity Units (WCUs) are units that measure my DynamoDB table's performance. DynamoDB pricing is based on RCUs and WCUs, so the more RCUs/WCUs consumed, the higher the cost of the database.

Amazon DynamoDB's Free Tier covers 25 RCUs and WCUs per month. I turned off auto-scaling to avoid potential higher charges, as DynamoDB automatically scales up operations, which would increase the consumption of RCU/WCUs, leading to additional cost.

Read/write capacity settings [Info](#)

Capacity mode

☒ **Provisioned**
Manage and optimize your costs by allocating read/write capacity in advance.

☐ **On-demand**
Simplify billing by paying for the actual reads and writes your application performs.

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.

☐ On

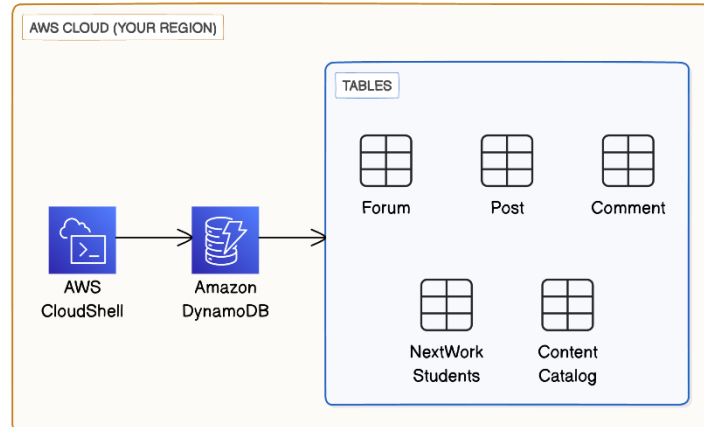
☒ **Off**

Provisioned capacity units

5



Using CLI and CloudShell



AWS CloudShell is an environment that lets us run code to interact with our AWS resources and services. Using AWS CloudShell is handy because the AWS CLI is already installed, so we can run commands straight away.

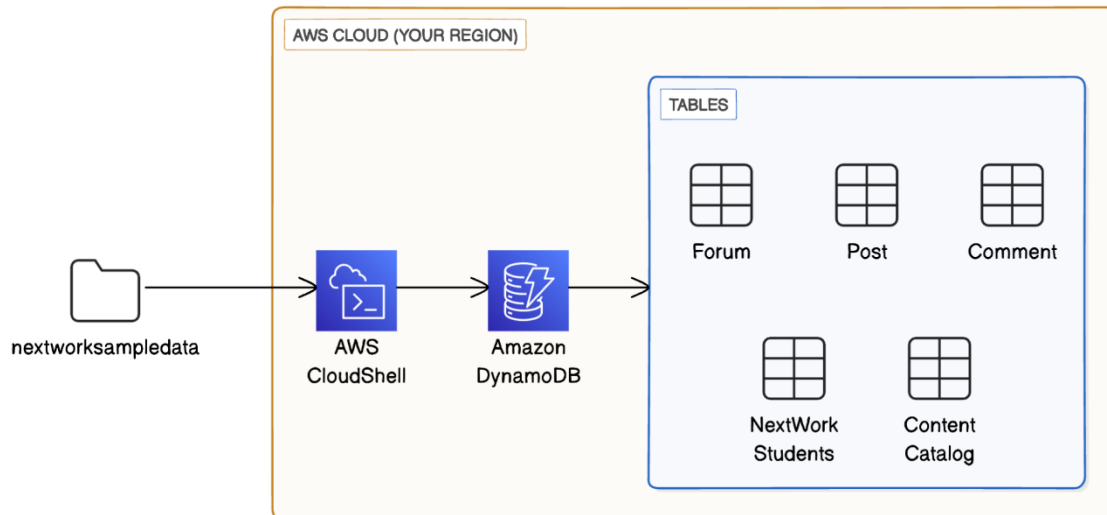
AWS CLI is a software that lets us interact with resources using commands instead of clicks in the console. This is a very practical tool that is preferred over using the AWS Management Console in everyday engineering operations in the real world.

I ran a CLI command in AWS CloudShell that created four new DynamoDB tables. The AWS CLI command is called `aws dynamodb create-table`, and I could even define the table name and its attributes all within the command.

```
CloudShell
ap-southeast-2 +
> --table-name Post \
> --query "TableDescription.TableStatus"
"CREATING"
[cloudshell-user@ip-10-134-57-4 ~]$ 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"
[cloudshell-user@ip-10-134-57-4 ~]$ 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"
```



Loading Data with CLI



I ran a CLI command in AWS CloudShell that load multiple pieces of data (i.e. load multiple items) into DynamoDb table I set up in the previous step. This AWS CLI command is structured as 'aws dynamodb batch-write-item --request-items file:// xyz'

```
}
[cloudshell-user@ip-10-134-57-4 nextworksampladata]$ aws dynamodb batch-write-item --request-items file:///ContentCatalog.json
{
  "UnprocessedItems": {}
}
[cloudshell-user@ip-10-134-57-4 nextworksampladata]$
[cloudshell-user@ip-10-134-57-4 nextworksampladata]$ aws dynamodb batch-write-item --request-items file:///Forum.json
{
  "UnprocessedItems": {}
}
[cloudshell-user@ip-10-134-57-4 nextworksampladata]$
[cloudshell-user@ip-10-134-57-4 nextworksampladata]$ aws dynamodb batch-write-item --request-items file:///Post.json
{
  "UnprocessedItems": {}
}
[cloudshell-user@ip-10-134-57-4 nextworksampladata]$
[cloudshell-user@ip-10-134-57-4 nextworksampladata]$ aws dynamodb batch-write-item --request-items file:///Comment.json
{
  "UnprocessedItems": {}
}
[cloudshell-user@ip-10-134-57-4 nextworksampladata]$
```



Observing Item Attributes

DynamoDB > Explore items: ContentCatalog > Edit item

Edit item Form JSON view

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](#)

Attributes Add new attribute ▼

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

I checked a ContentCatalog item, which had the following attributes: ID, Authors, Content Type, Difficulty Levels, Price, Project Category, Published, Title, and URL.

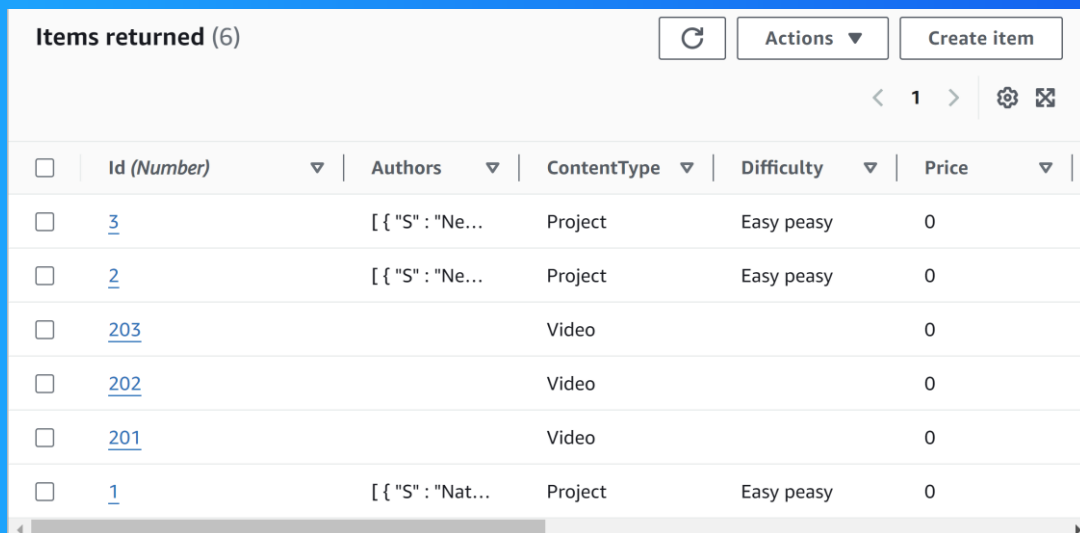
I checked another ContentCatalog item, which had a different set of attributes: Services, Title, Video type etc.



Benefits of DynamoDB

A benefit of DynamoDB over relational databases is its flexibility, as items can have their own set of attributes. This is great for scenarios where a table contains different types of data, and some attributes don't apply to all data in that table.

Another benefit over relational databases is speed, as DynamoDB can quickly partition its data to narrow down search using partition keys. This is faster than RDS, which needs to scan entire tables to find specific pieces of data.



Items returned (6)					Refresh	Actions ▼	Create item
	Id (Number) ▼	Authors ▼	ContentType ▼	Difficulty ▼	Price ▼		
<input type="checkbox"/>	3	[{ "S" : "Ne...	Project	Easy peasy	0		
<input type="checkbox"/>	2	[{ "S" : "Ne...	Project	Easy peasy	0		
<input type="checkbox"/>	203		Video		0		
<input type="checkbox"/>	202		Video		0		
<input type="checkbox"/>	201		Video		0		
<input type="checkbox"/>	1	[{ "S" : "Nat...	Project	Easy peasy	0		



Everyone should be in a job they love.

Check out nextwork.org for
more projects

