



Encrypt Data with AWS KMS

GA

gazzok@gmail.com

The screenshot shows the Amazon DynamoDB console with the path: DynamoDB > Explore Items > nextwork-kms-table. On the left, the sidebar shows the 'DynamoDB' section with options like Dashboard, Tables, and Explore items. The main area displays a table named 'nextwork-kms-table' with one item. A blue header bar at the top says 'Share your feedback on Amazon DynamoDB' with a note: 'Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.' Below the table, there's a 'Scan or query items' button and a 'Find tables' search bar. At the bottom of the page, an 'Access denied' error message is displayed in a red-bordered box:

Access denied
You don't have permission to `kms:Decrypt`. To request access, copy the following text and send it to your AWS administrator. Learn more about troubleshooting access denied errors.

User: arn:aws:iam:314146329097:user/nextwork-kms-user
Action: kms:Decrypt
On resource(s): arn:aws:kms:eu-west-2:314146329097:key/e1768d51-b3bc-446e-a13f-679eb02628da
Context: no identity-based policy allows the action

Introducing Today's Project!

In this project, I will demonstrate the use of KMS keys to encrypt a database. The goal is to protect the data of the database through the use of granting permission to users for KMS keys.

Tools and concepts

Services I used include DynamoDB, IAM and AWS KMS. Key concepts I learnt include encryption, permission access and database creation.

Project reflection

This project took me approximately 1.5 hours. It was rewarding to see the power of AWS KMS keys on securing data in a database.

I did this project to improve my abilities on security in AWS. This was a great project that definitely met my expectations.

Encryption and KMS

Encryption is the process of using algorithms to convert data into a secure format called ciphertext. Companies and developers do this to protect your data. Encryption keys are essentially decoders to get the encrypted ciphertext back to normal.

AWS Key Management Service (KMS) is a secure vault for your encryption keys. Key management systems are important because it helps you manage all your keys in one place. KMS has the benefit of also giving you logs on every time a key is used.

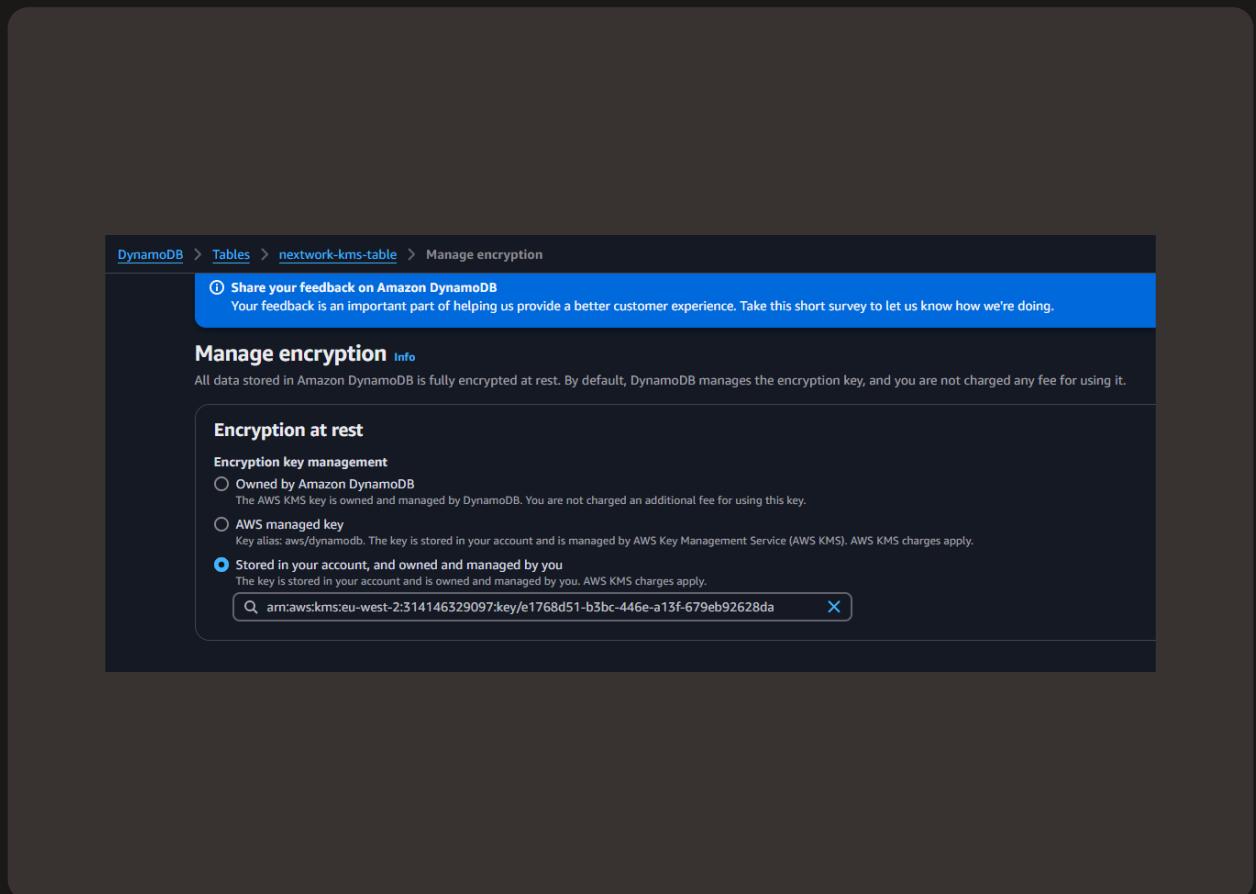
Encryption keys are broadly categorized as the algorithm that will decrypt & encrypt your data. I set up a symmetric key because they are generally faster and more efficient for encrypting large amounts of data (and I'm not sharing the key).

The screenshot shows the AWS KMS console interface. At the top, there is a green success message: "Success Your AWS KMS key was created with alias nextwork-kms-key and key ID e1768d51-b3bc-446e-a13f-679eb92628da." Below this, a section titled "Customer managed keys (1)" is displayed. A search bar labeled "Filter keys by properties or tags" is present. The table below has columns for "Aliases", "Key ID", "Status", and "Key type". One row is shown, corresponding to the successfully created key: "nextwork-kms-key" (Alias), "e1768d51-b3bc-446e-a13f-679eb92628da..." (Key ID), "Enabled" (Status), and "Symmetric" (Key type). The entire screenshot is framed by a dark gray border.

Encrypting Data

My encryption key will safeguard data in DynamoDB, which is at rest

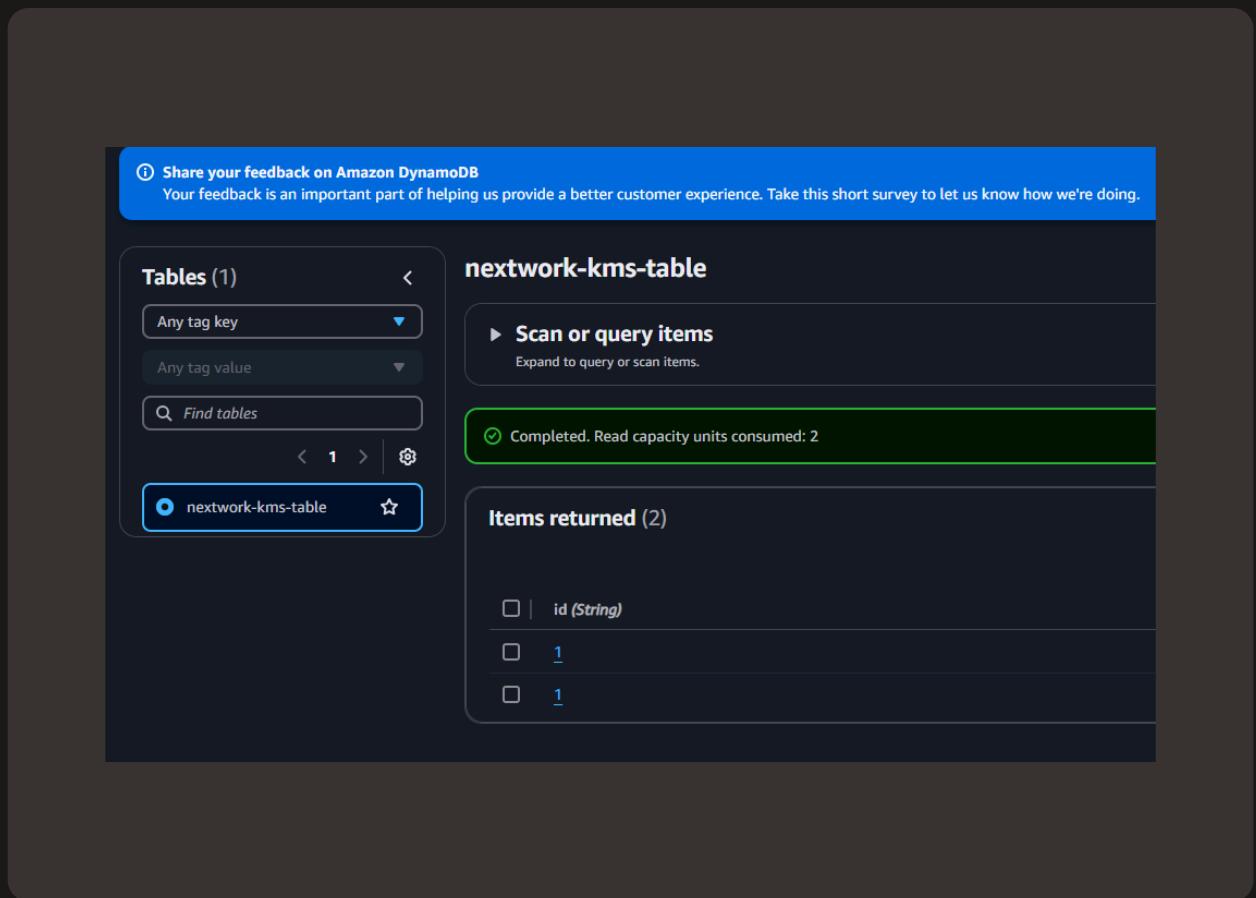
The different encryption options in DynamoDB include Owned by Amazon DynamoDB, AWS managed key and Stored in your account, and owned and managed by you. Their differences are based on ownership & management of the keys.



Data Visibility

Rather than controlling who has access to the key, KMS manages user permissions by who has been granted the permission to use the key.

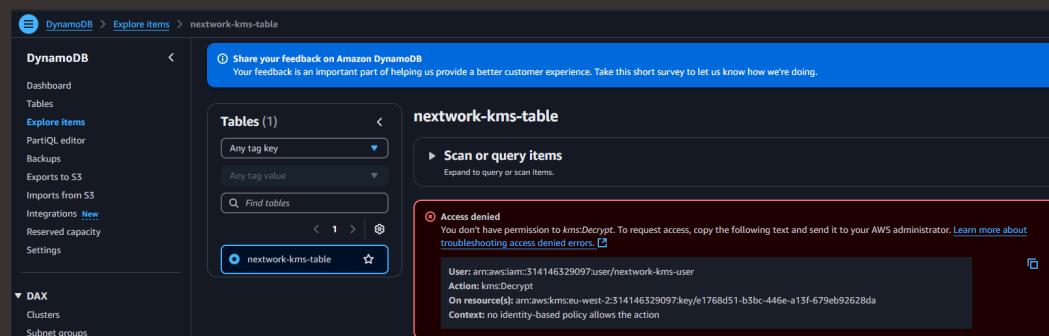
Despite encrypting my DynamoDB table, I could still see the table's items because my IAM user has permissions to use the encryption key in KMS. DynamoDB uses transparent data encryption, which means an authorized user will see it in a decrypted format



Denying Access

I configured a new IAM user to verify our encryption using KMS is working correctly. The permission policies I granted this user are full access to DynamoDB, but not to my KMS key.

After accessing the DynamoDB table as the test user, I encountered an 'Access Denied' error because this user does not have KMS permissions and cannot decrypt the data in my DynamoDB table. This confirmed my KMS key is working how it should.



EXTRA: Granting Access

To let my test user use the encryption key, I granted the user permission to use the KMS key. My key's policy was updated to allow the new user permission to use it as seen in the code.

Using the test user, I retried accessing the DynamoDB table. I observed that the operation was completed this time which confirmed that adding the permission to use the KMS key has worked.

Encryption secures data instead of access to a resource (e.g. a DynamoDB table) or an entire service (e.g. DynamoDB). I could combine encryption with the afore mentioned measures to further limit access and keep our data & resources secure.

```
Key policy
34      "kms:CancelKeyDeletion",
35      "kms:RotateKeyOnDemand"
36    ],
37    "Resource": "*"
38  },
39  {
40    "Sid": "Allow use of the key",
41    "Effect": "Allow",
42    "Principal": [
43      "AWS": [
44        "arn:aws:iam::314146329097:user/Garyking-IAM-Admin",
45        "arn:aws:iam::314146329097:user/nextwork-kms-user"
46      ]
47    },
48    "Action": [
49      "kms:Encrypt",
50      "kms:Decrypt",
51      "kms:ReEncrypt",
52      "kms:GenerateDataKey",
53      "kms:DescribeKey"
```

**Everyone
should be in a
job they love.**

Check out nextwork.org for
more projects

