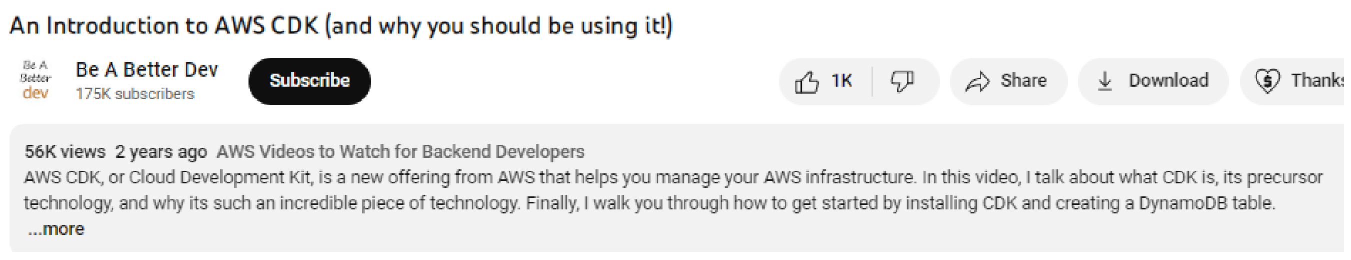


An Introduction to AWS CDK (and why you should be using it!)

Be A Better Dev
175K subscribers

Subscribe

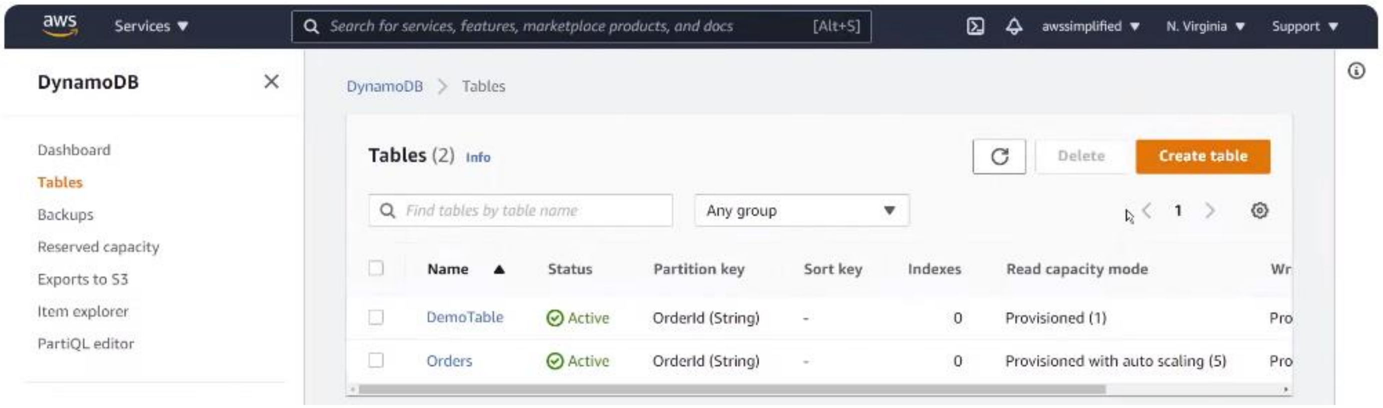
1K | Share | Download | Thanks

56K views 2 years ago AWS Videos to Watch for Backend Developers
AWS CDK, or Cloud Development Kit, is a new offering from AWS that helps you manage your AWS infrastructure. In this video, I talk about what CDK is, its precursor technology, and why its such an incredible piece of technology. Finally, I walk you through how to get started by installing CDK and creating a DynamoDB table.
...more

Cloud
Development
Kit

CREATE
CONFIGURE
DEPLOY

Manually?
CloudFormation?

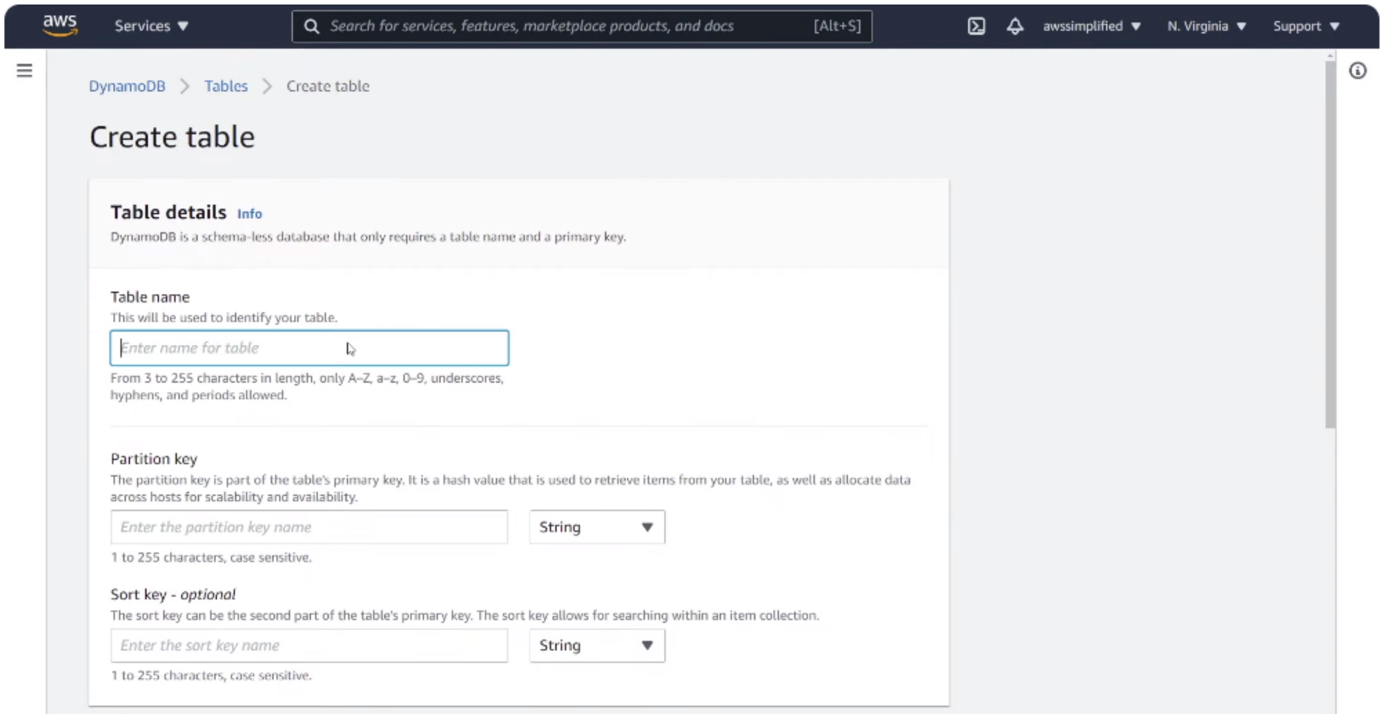


Tables (2) Info

Delete | Create table

Find tables by table name | Any group | 1

Name ▲	Status	Partition key	Sort key	Indexes	Read capacity mode	Wr
DemoTable	✓ Active	OrderId (String)	-	0	Provisioned (1)	Pro
Orders	✓ Active	OrderId (String)	-	0	Provisioned with auto scaling (5)	Pro



aws
Services ▼
Search for services, features, marketplace products, and docs [Alt+S]
awssimplified ▼
N. Virginia ▼
Support ▼
DynamoDB > Tables > Create table

Create table

Table details Info

DynamoDB is a schema-less database that only requires a table name and a primary key.

Table name
This will be used to identify your table.
Enter name for table
From 3 to 255 characters in length, only A-Z, a-z, 0-9, underscores, hyphens, and periods allowed.

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table, as well as allocate data across hosts for scalability and availability.
Enter the partition key name | String ▼
1 to 255 characters, case sensitive.

Sort key - *optional*
The sort key can be the second part of the table's primary key. The sort key allows for searching within an item collection.
Enter the sort key name | String ▼
1 to 255 characters, case sensitive.

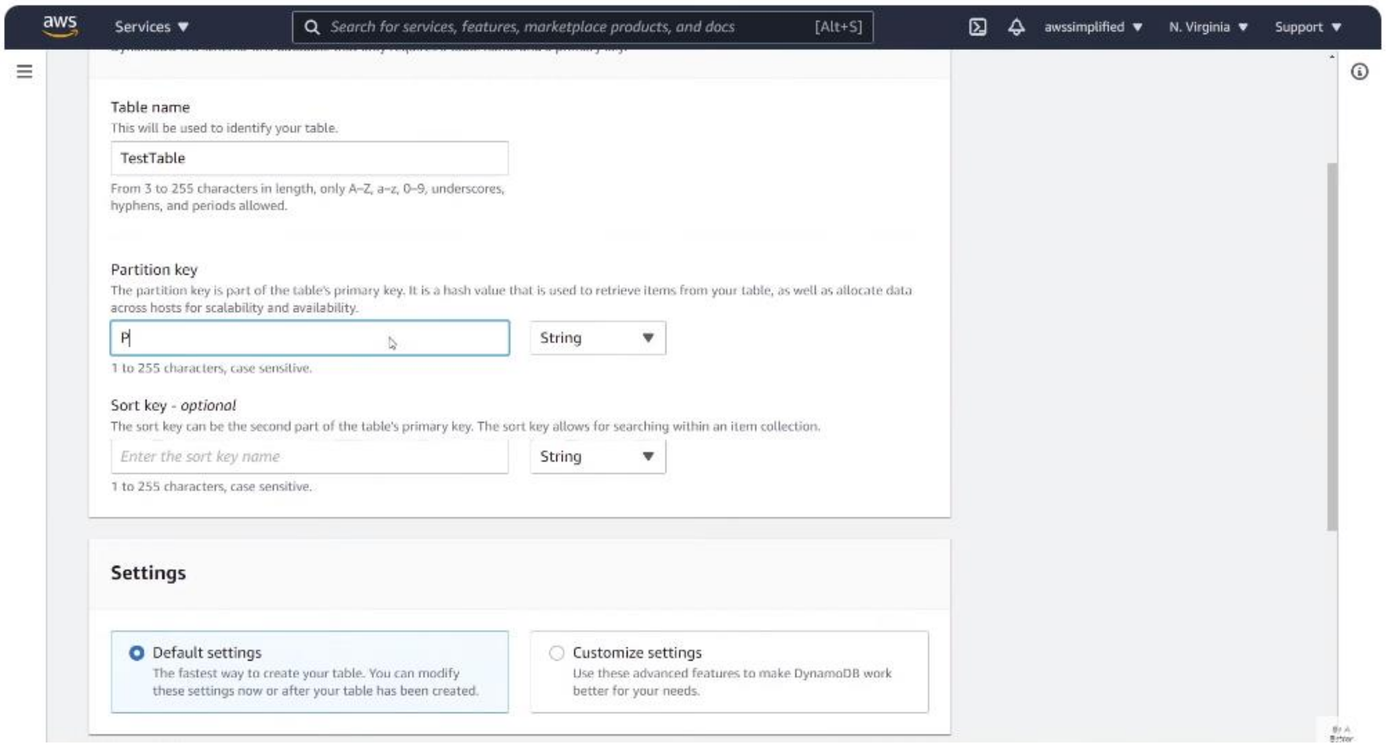

aws
Services ▼
Search for services, features, marketplace products, and docs [Alt+S]
awssimplified ▼
N. Virginia ▼
Support ▼

Table name
This will be used to identify your table.

TestTable

From 3 to 255 characters in length, only A–Z, a–z, 0–9, underscores, hyphens, and periods allowed.

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table, as well as allocate data across hosts for scalability and availability.

P | String

1 to 255 characters, case sensitive.

Sort key - *optional*
The sort key can be the second part of the table's primary key. The sort key allows for searching within an item collection.

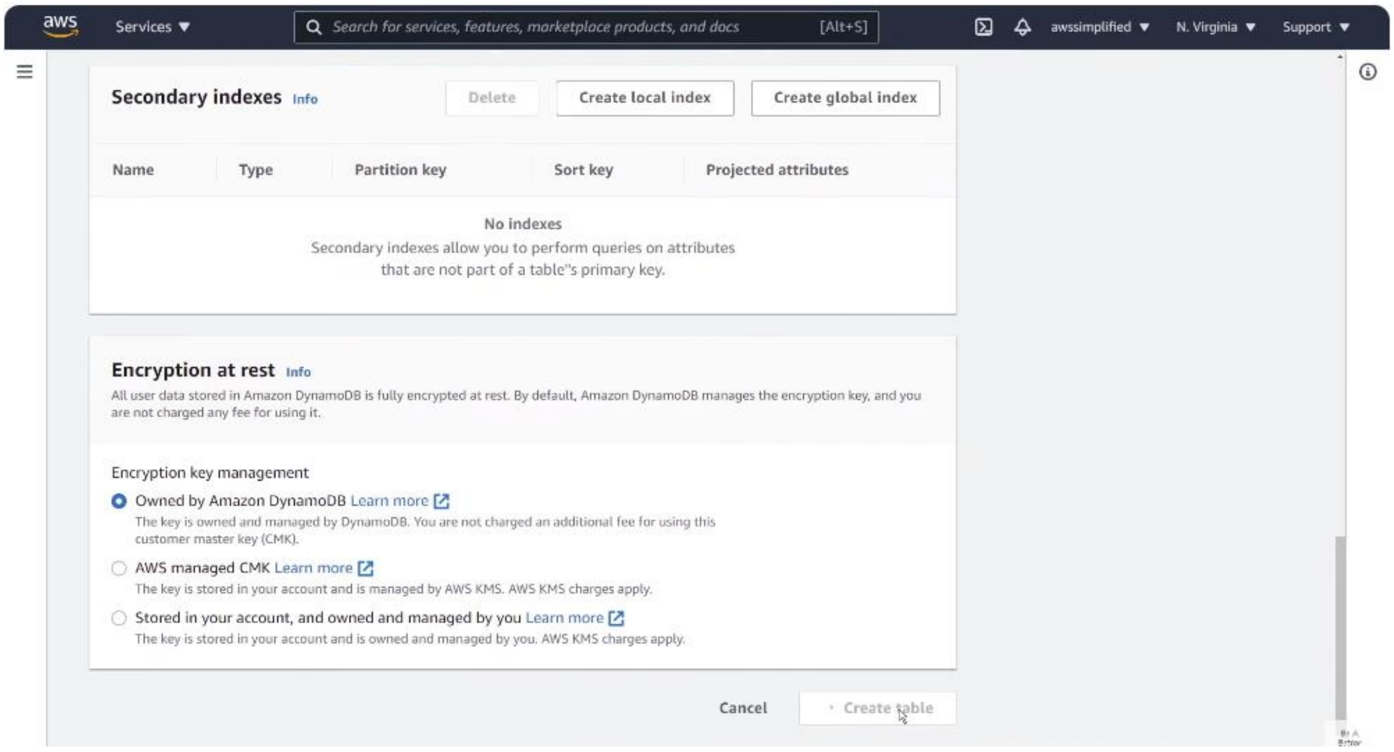
Enter the sort key name | String

1 to 255 characters, case sensitive.

Settings

[✓ Default settings] The fastest way to create your table. You can modify these settings now or after your table has been created.

Customize settings — Use these advanced features to make DynamoDB work better for your needs.


aws
Services ▼
Search for services, features, marketplace products, and docs [Alt+S]
awssimplified ▼
N. Virginia ▼
Support ▼

Secondary indexes Info

Name	Type	Partition key	Sort key	Projected attributes

No indexes
Secondary indexes allow you to perform queries on attributes that are not part of a table''s primary key.

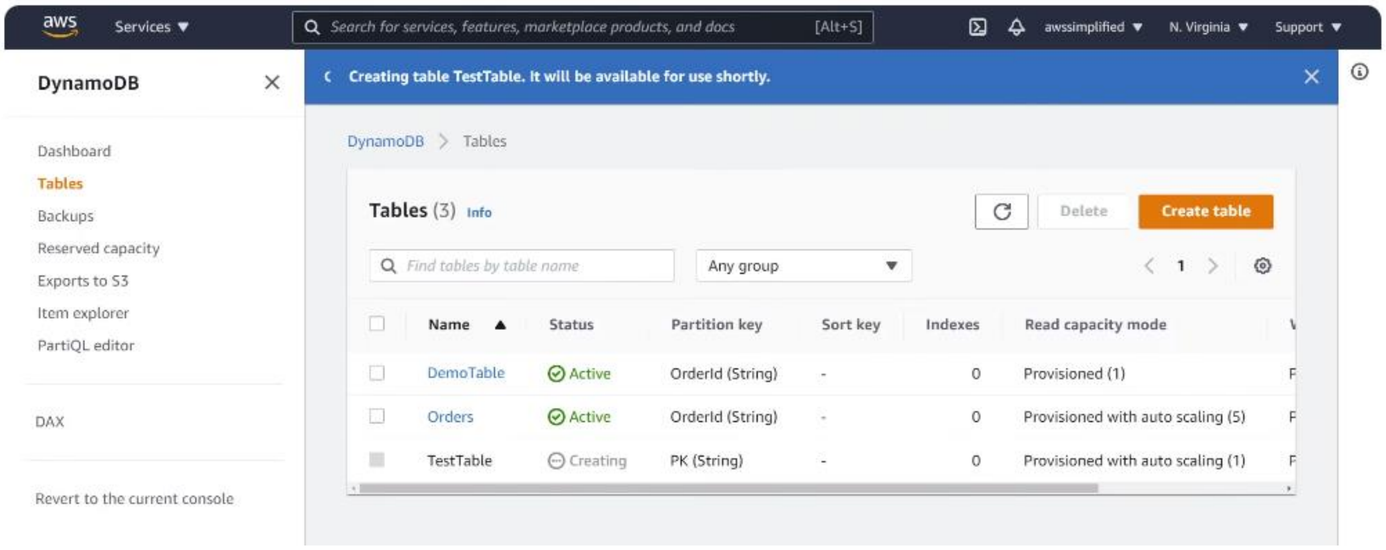
Encryption at rest Info
All user data stored in Amazon DynamoDB is fully encrypted at rest. By default, Amazon DynamoDB manages the encryption key, and you are not charged any fee for using it.

Encryption key management

[✓ Owned by Amazon DynamoDB] Learn more
The key is owned and managed by DynamoDB. You are not charged an additional fee for using this customer master key (CMK).

AWS managed CMK Learn more
The key is stored in your account and is managed by AWS KMS. AWS KMS charges apply.

Stored in your account, and owned and managed by you Learn more
The key is stored in your account and is owned and managed by you. AWS KMS charges apply.



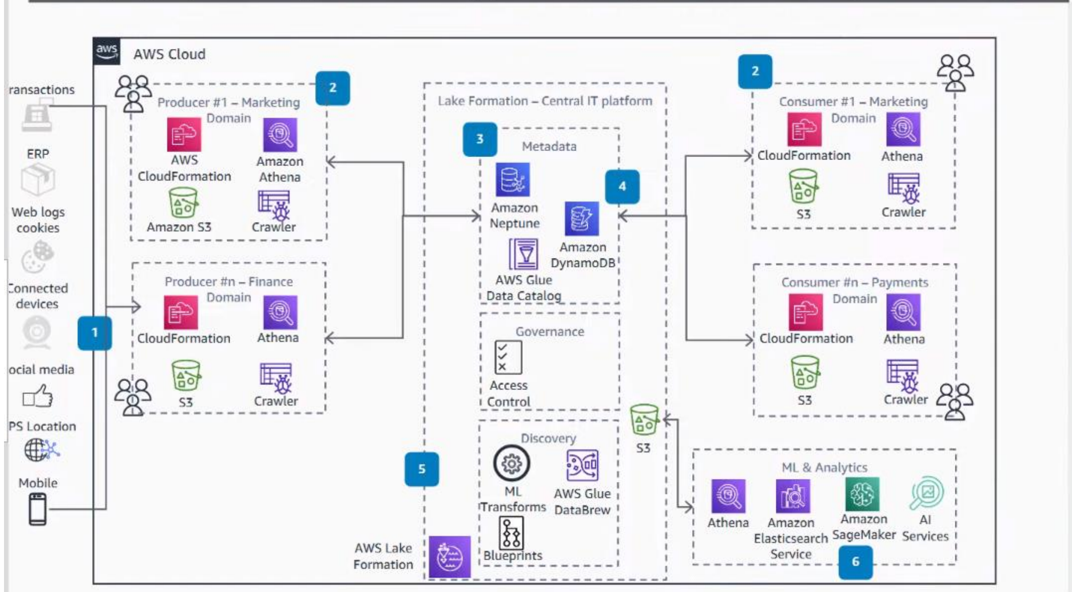
Creating table TestTable. It will be available for use shortly.

Tables (3) Info

	Name ▲	Status	Partition key	Sort key	Indexes	Read capacity mode
☐	DemoTable	✓ Active	OrderId (String)	-	0	Provisioned (1)
☐	Orders	✓ Active	OrderId (String)	-	0	Provisioned with auto scaling (5)
☐	TestTable	Creating	PK (String)	-	0	Provisioned with auto scaling (1)

Consumer Packaged Goods Data Mesh

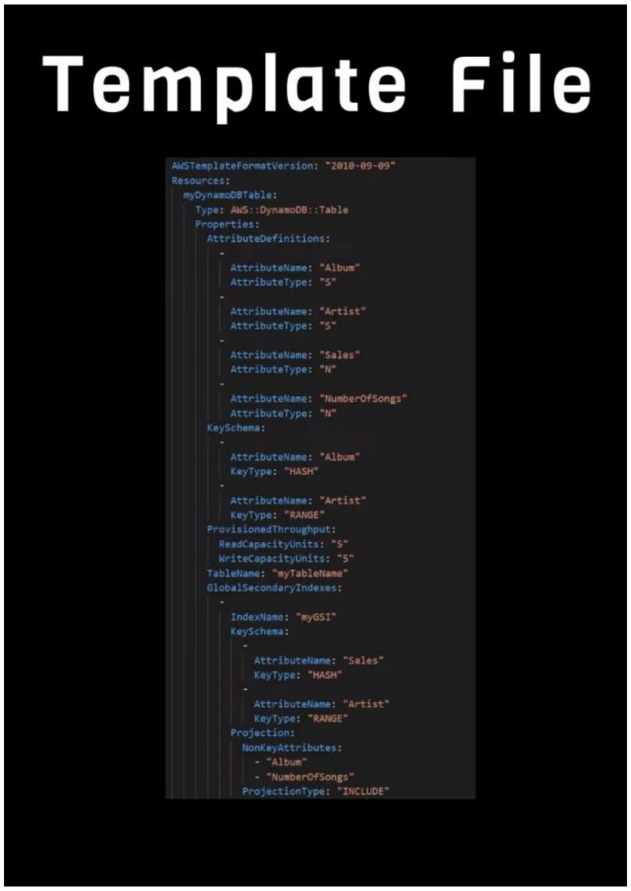
Transitioning to online selling accelerated the amount of data generated and maintained by business operations. This new need for generating, accessing, and analyzing data at scale presents new data management challenges. AWS proposes this new approach following the principles of flexibility, domain-owned design, maintaining data properties, and context throughout the data lifecycle.



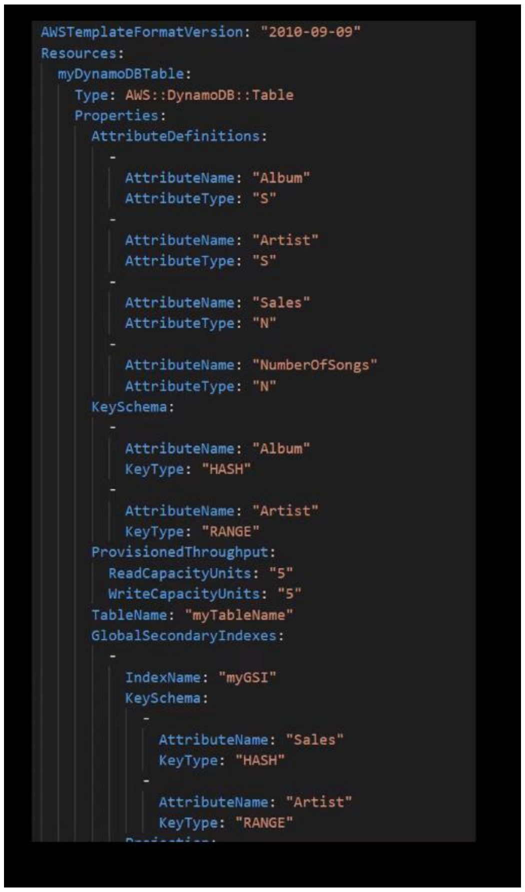
1. Data is ingested into AWS using batch processing, near real time, streaming and SSH File Transfer Protocol (SFTP).
2. Data sources are managed by the business domain. Consumers and producers use organization-level blueprints, providing core services like security, governance, IAM roles, and standards.
3. Metadata is managed via multiple services. **AWS Glue** is used for data cataloging. Data lineage is stored in **Amazon Neptune**. Data contracts are stored in **Amazon DynamoDB**.
4. Consumer search the data using the contract's properties
5. **AWS Lake Formation** provides centralized management of fine-grained permissions. It also enables automatic schema discovery and conversion to the required format
6. Machine learning (ML) and analytics can be performed thru **Lake Formation**. This new mesh node enables insights and strategy.

aws

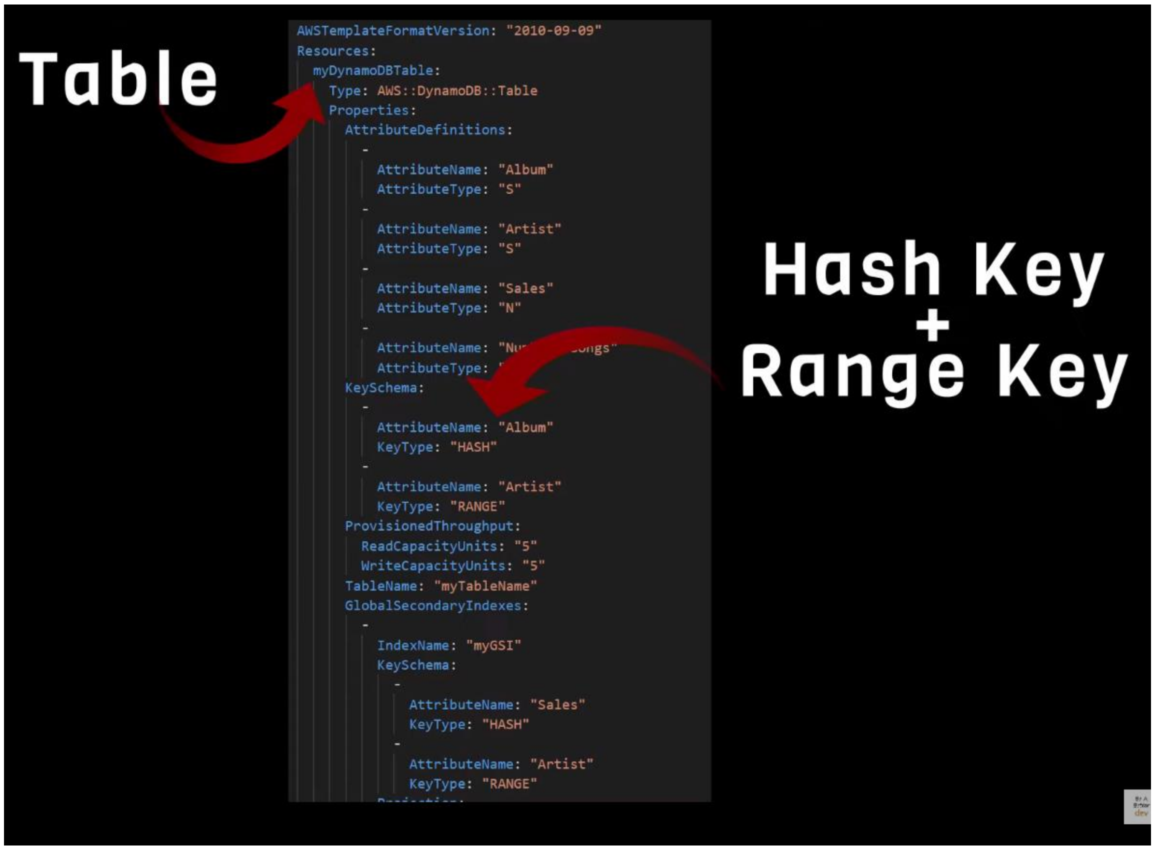
AWS Reference Architecture



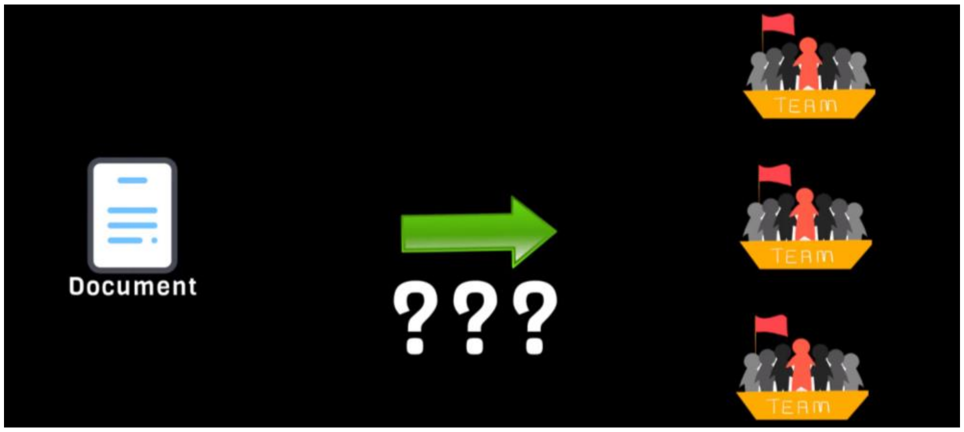
```yaml
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  myDynamoDBTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        -
          AttributeName: "Album"
          AttributeType: "S"
        -
          AttributeName: "Artist"
          AttributeType: "S"
        -
          AttributeName: "Sales"
          AttributeType: "N"
        -
          AttributeName: "NumberOfSongs"
          AttributeType: "N"
      KeySchema:
        -
          AttributeName: "Album"
          KeyType: "HASH"
        -
          AttributeName: "Artist"
          KeyType: "RANGE"
      ProvisionedThroughput:
        ReadCapacityUnits: "5"
        WriteCapacityUnits: "5"
      TableName: "myTableName"
      GlobalSecondaryIndexes:
        -
          IndexName: "myGSI"
          KeySchema:
            -
              AttributeName: "Sales"
              KeyType: "HASH"
            -
              AttributeName: "Artist"
              KeyType: "RANGE"
          Projection:
            NonKeyAttributes:
              - "Album"
              - "NumberOfSongs"
            ProjectionType: "INCLUDE"
```

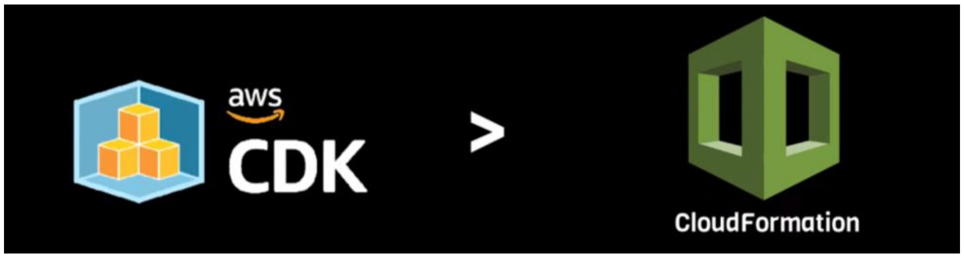


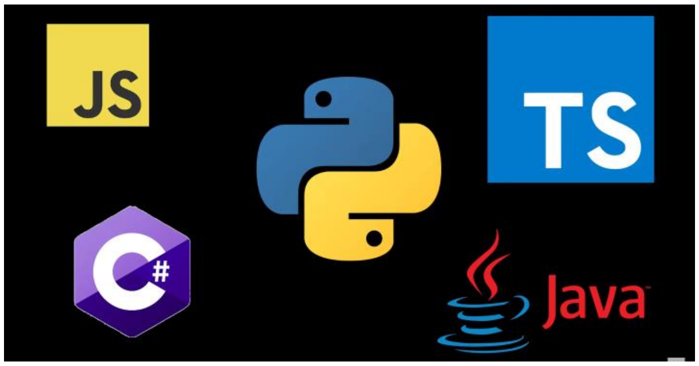
```yaml
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  myDynamoDBTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        -
          AttributeName: "Album"
          AttributeType: "S"
        -
          AttributeName: "Artist"
          AttributeType: "S"
        -
          AttributeName: "Sales"
          AttributeType: "N"
        -
          AttributeName: "NumberOfSongs"
          AttributeType: "N"
      KeySchema:
        -
          AttributeName: "Album"
          KeyType: "HASH"
        -
          AttributeName: "Artist"
          KeyType: "RANGE"
      ProvisionedThroughput:
        ReadCapacityUnits: "5"
        WriteCapacityUnits: "5"
      TableName: "myTableName"
      GlobalSecondaryIndexes:
        -
          IndexName: "myGSI"
          KeySchema:
            -
              AttributeName: "Sales"
              KeyType: "HASH"
            -
              AttributeName: "Artist"
              KeyType: "RANGE"
```

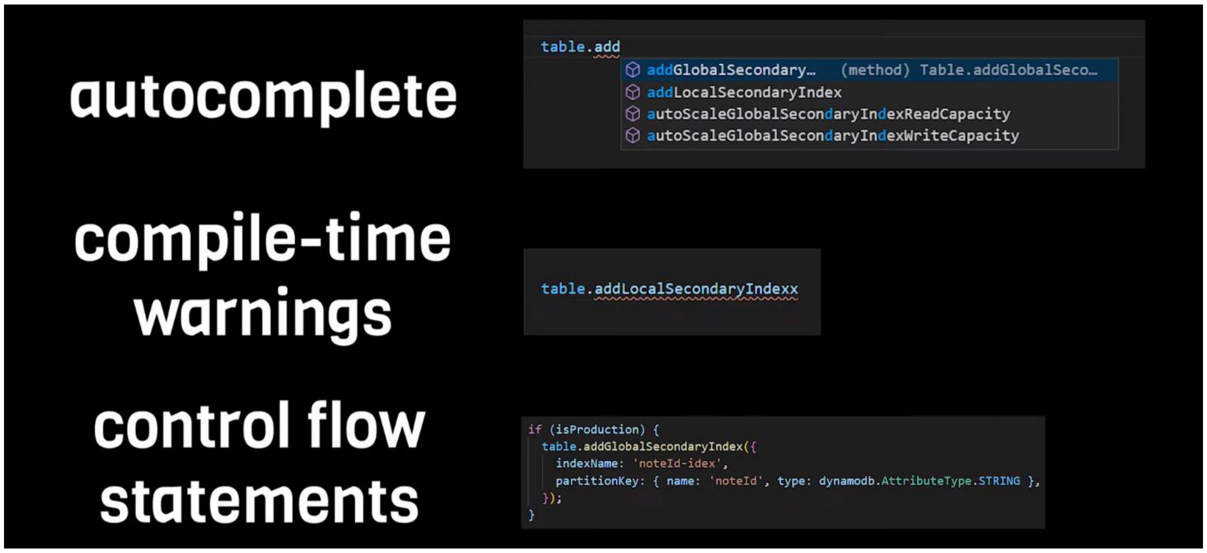


```yaml
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  myDynamoDBTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        -
          AttributeName: "Album"
          AttributeType: "S"
        -
          AttributeName: "Artist"
          AttributeType: "S"
        -
          AttributeName: "Sales"
          AttributeType: "N"
        -
          AttributeName: "Nu      ongs"
          AttributeType: "
      KeySchema:
        -
          AttributeName: "Album"
          KeyType: "HASH"
        -
          AttributeName: "Artist"
          KeyType: "RANGE"
      ProvisionedThroughput:
        ReadCapacityUnits: "5"
        WriteCapacityUnits: "5"
      TableName: "myTableName"
      GlobalSecondaryIndexes:
        -
          IndexName: "myGSI"
          KeySchema:
            -
              AttributeName: "Sales"
              KeyType: "HASH"
            -
              AttributeName: "Artist"
              KeyType: "RANGE"
```








autocomplete

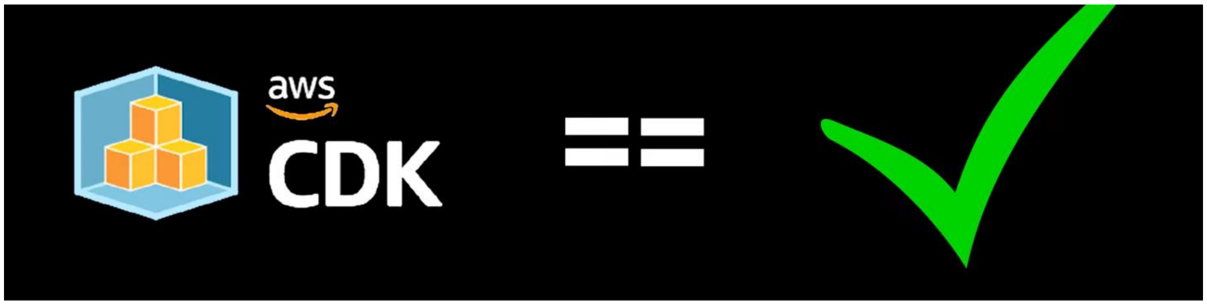
```
table.add
  addGlobalSecondary…    (method) Table.addGlobalSeco…
  addLocalSecondaryIndex
  autoScaleGlobalSecondaryIndexReadCapacity
  autoScaleGlobalSecondaryIndexWriteCapacity
```

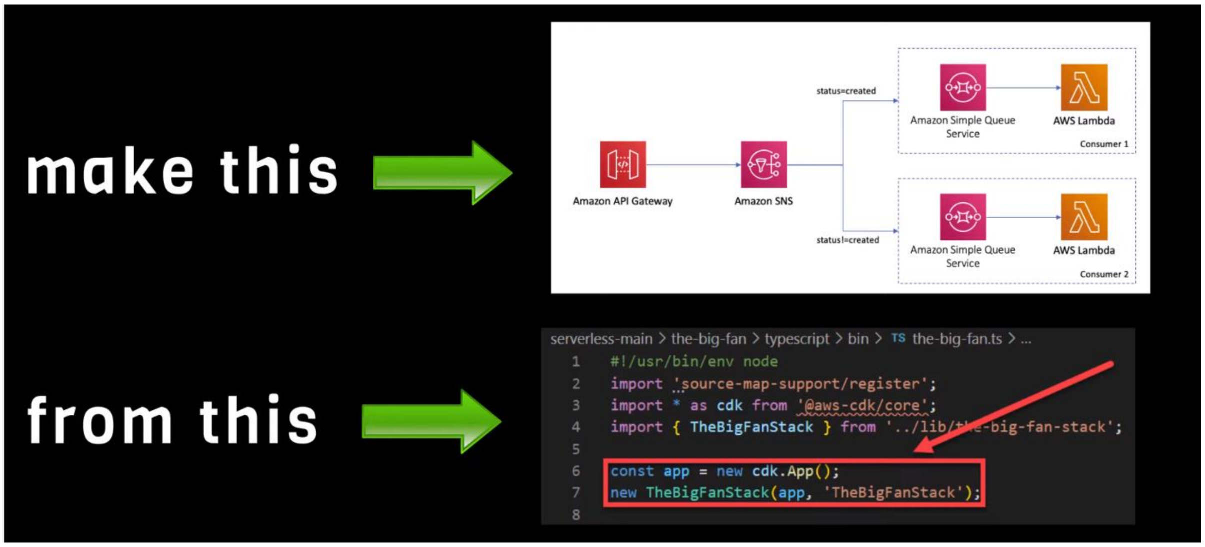
compile-time warnings

```
table.addLocalSecondaryIndexx
```

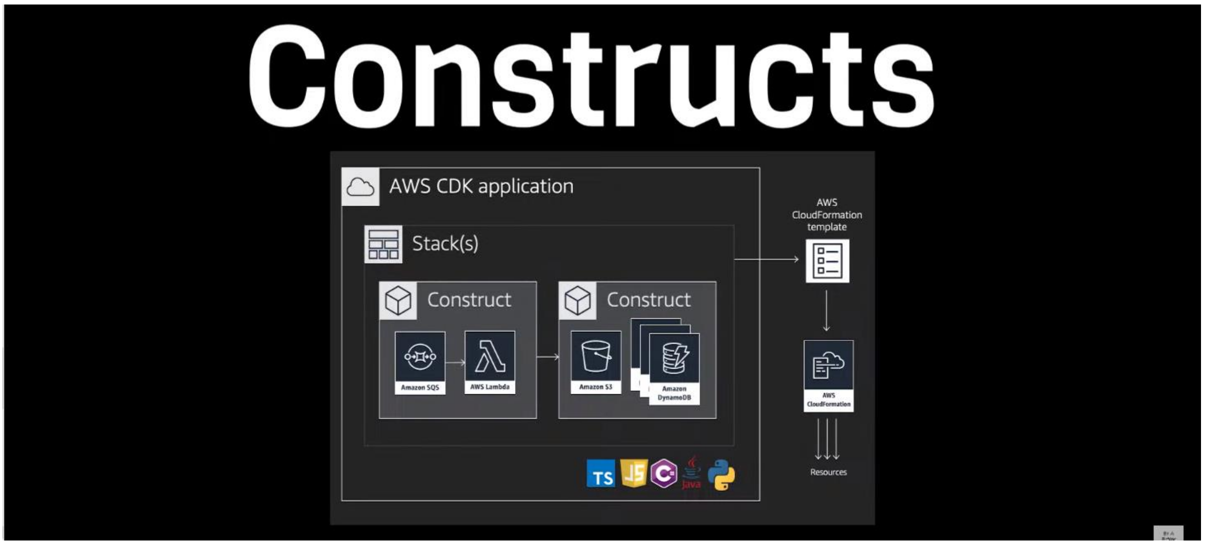
control flow statements

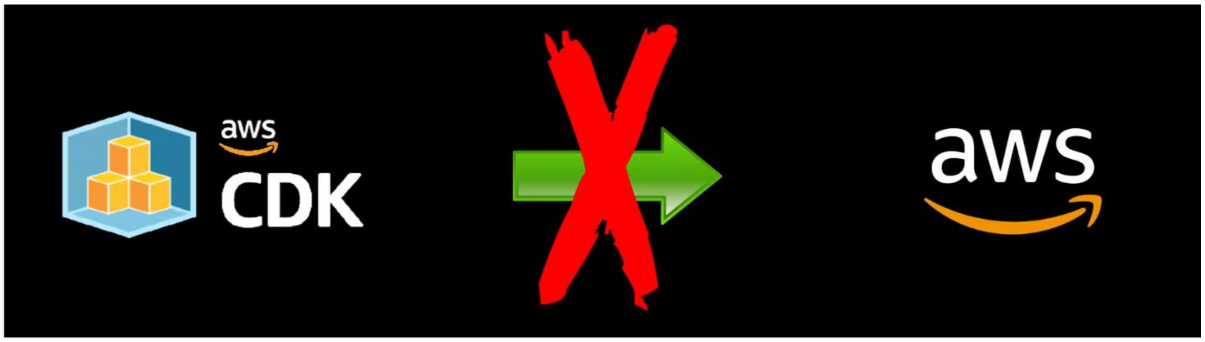
```
if (isProduction) {
  table.addGlobalSecondaryIndex({
    indexName: 'noteId-idex',
    partitionKey: { name: 'noteId', type: dynamodb.AttributeType.STRING },
  });
}
```

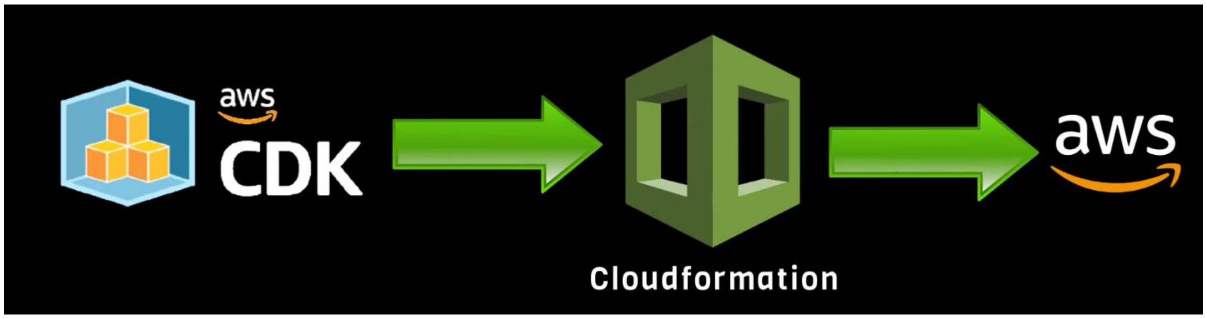


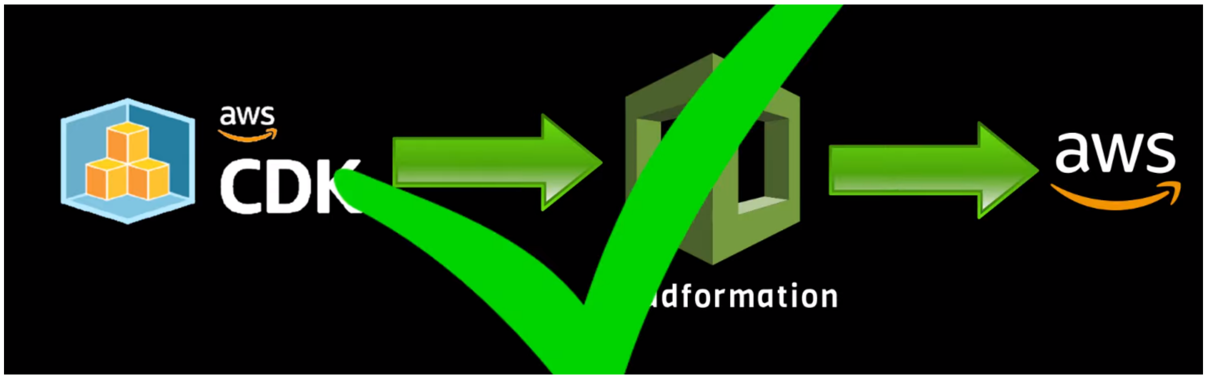

make this
from this
status=created
status!=created
Amazon API Gateway
Amazon SNS
Amazon Simple Queue Service
AWS Lambda
Consumer 1
Amazon Simple Queue Service
AWS Lambda
Consumer 2
serverless-main > the-big-fan > typescript > bin > TS the-big-fan.ts > ...

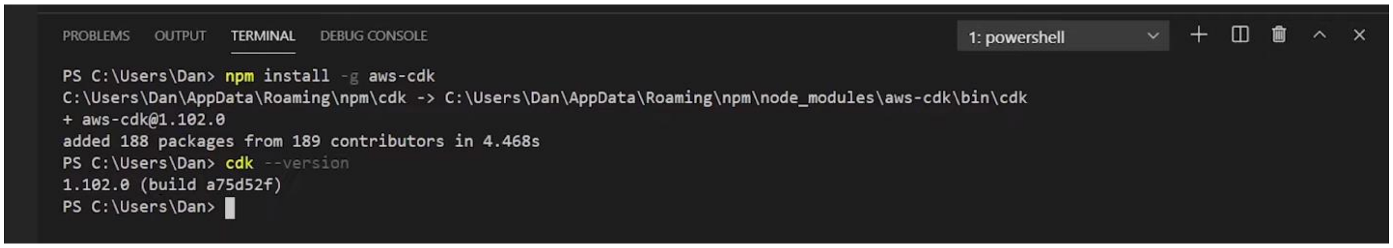
```
#!/usr/bin/env node
import 'source-map-support/register';
import * as cdk from '@aws-cdk/core';
import { TheBigFanStack } from '../lib/the-big-fan-stack';

const app = new cdk.App();
new TheBigFanStack(app, 'TheBigFanStack');

```



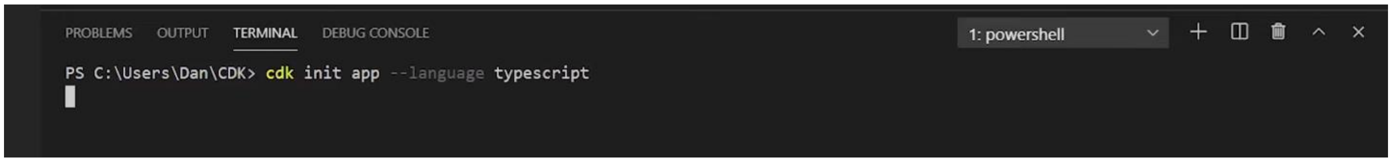






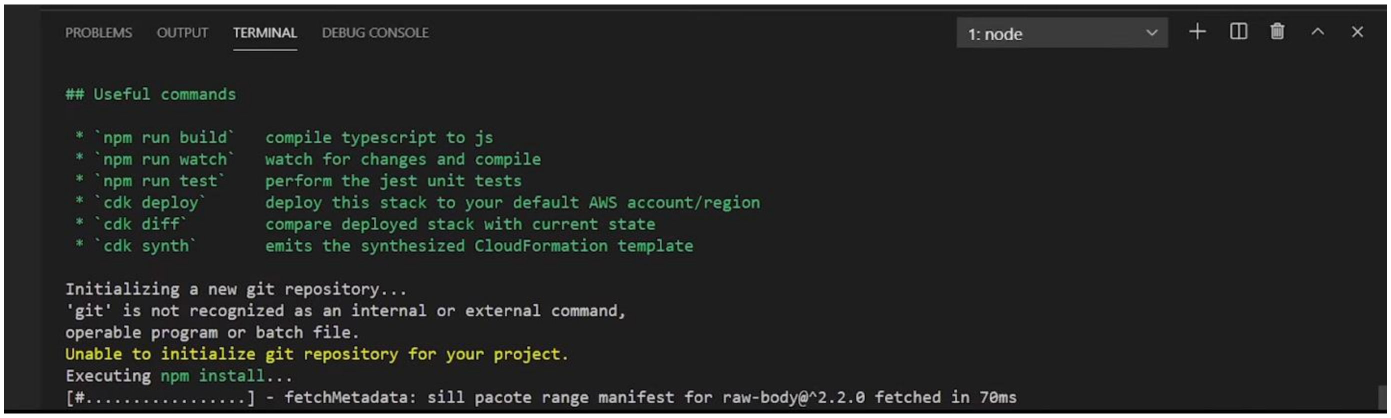


```
PS C:\Users\Dan> npm install -g aws-cdk
C:\Users\Dan\AppData\Roaming\npm\cdk -> C:\Users\Dan\AppData\Roaming\npm\node_modules\aws-cdk\bin\cdk
+ aws-cdk@1.102.0
added 188 packages from 189 contributors in 4.468s
PS C:\Users\Dan> cdk --version
1.102.0 (build a75d52f)
PS C:\Users\Dan>
```

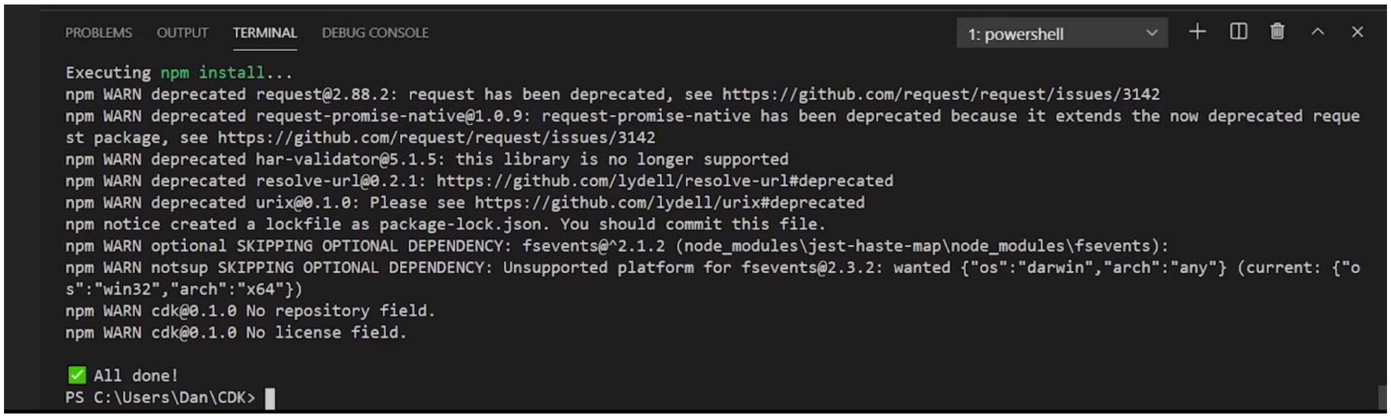


```
PS C:\Users\Dan\CDK> cdk init app --language typescript
```

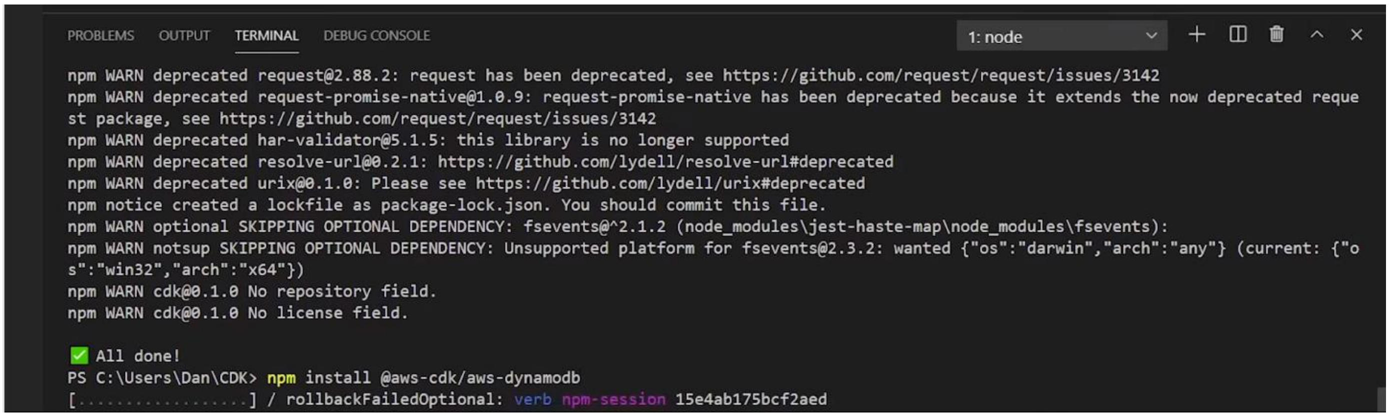
We can generate a sample initial project using the above command



```
## Useful commands

 * `npm run build`   compile typescript to js
 * `npm run watch`   watch for changes and compile
 * `npm run test`    perform the jest unit tests
 * `cdk deploy`      deploy this stack to your default AWS account/region
 * `cdk diff`        compare deployed stack with current state
 * `cdk synth`       emits the synthesized CloudFormation template

Initializing a new git repository...
'git' is not recognized as an internal or external command,
operable program or batch file.
Unable to initialize git repository for your project.
Executing npm install...
[#...............] - fetchMetadata: sill pacote range manifest for raw-body@^2.2.0 fetched in 70ms
```

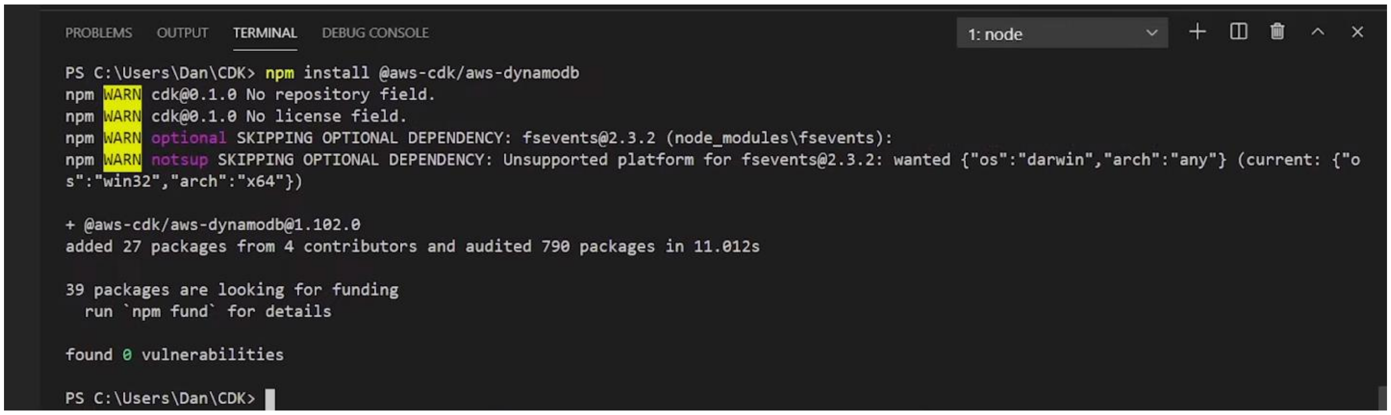


```
Executing npm install...
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated request-promise-native@1.0.9: request-promise-native has been deprecated because it extends the now deprecated reque
st package, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^2.1.2 (node_modules\jest-haste-map\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"o
s":"win32","arch":"x64"})
npm WARN cdk@0.1.0 No repository field.
npm WARN cdk@0.1.0 No license field.

✅ All done!
PS C:\Users\Dan\CDK>
```



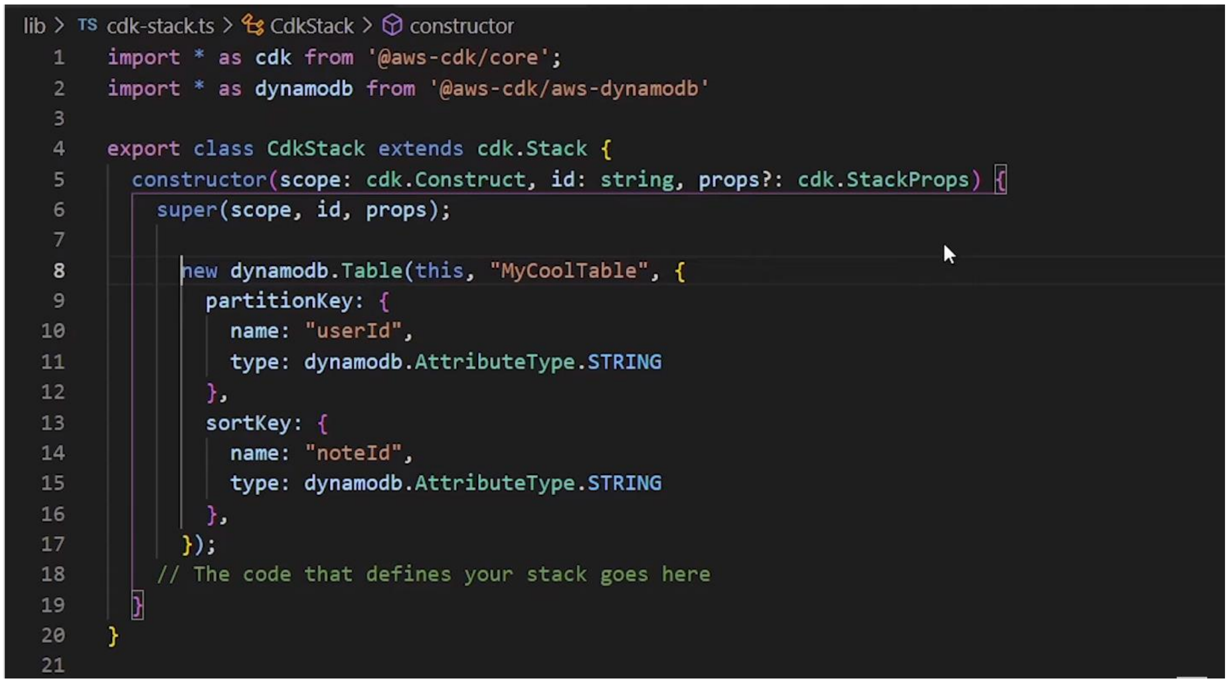
```
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated request-promise-native@1.0.9: request-promise-native has been deprecated because it extends the now deprecated reque
st package, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^2.1.2 (node_modules\jest-haste-map\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"o
s":"win32","arch":"x64"})
npm WARN cdk@0.1.0 No repository field.
npm WARN cdk@0.1.0 No license field.

✅ All done!
PS C:\Users\Dan\CDK> npm install @aws-cdk/aws-dynamodb
[..................] / rollbackFailedOptional: verb npm-session 15e4ab175bcf2aed
```

We then install an AWS construct that allows us to create a DynamoDB table.



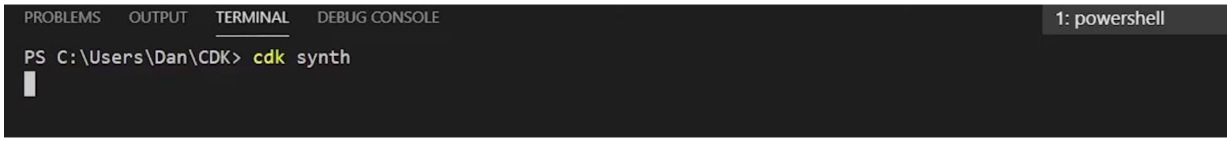
```
PS C:\Users\Dan\CDK> npm install @aws-cdk/aws-dynamodb
npm WARN cdk@0.1.0 No repository field.
npm WARN cdk@0.1.0 No license field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"o
s":"win32","arch":"x64"})

+ @aws-cdk/aws-dynamodb@1.102.0
added 27 packages from 4 contributors and audited 790 packages in 11.012s

39 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

PS C:\Users\Dan\CDK>
```

Now we are ready to write some code.



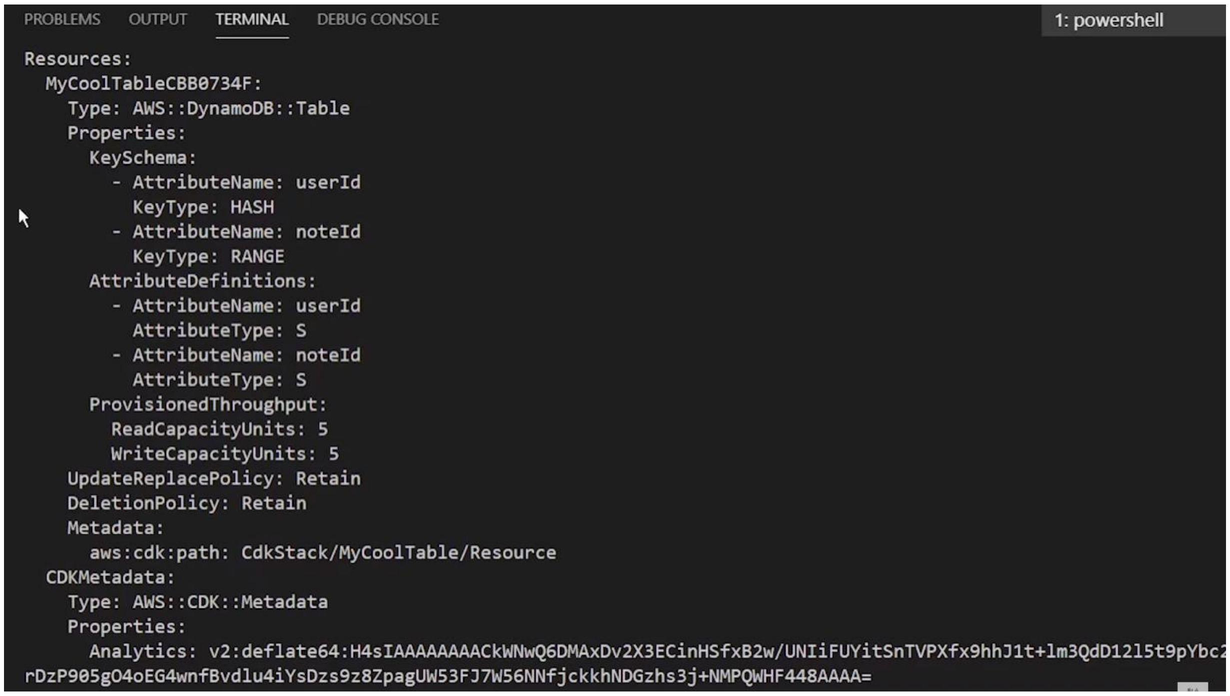
```typescript
import * as cdk from '@aws-cdk/core';
import * as dynamodb from '@aws-cdk/aws-dynamodb'

export class CdkStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    new dynamodb.Table(this, "MyCoolTable", {
      partitionKey: {
        name: "userId",
        type: dynamodb.AttributeType.STRING
      },
      sortKey: {
        name: "noteId",
        type: dynamodb.AttributeType.STRING
      },
    });
    // The code that defines your stack goes here
  }
}

```

With the CDK, notice that we aren't specifying a lot of information like when using CloudFormation. Using the CDK provides us with sensible constructs that we can leverage.

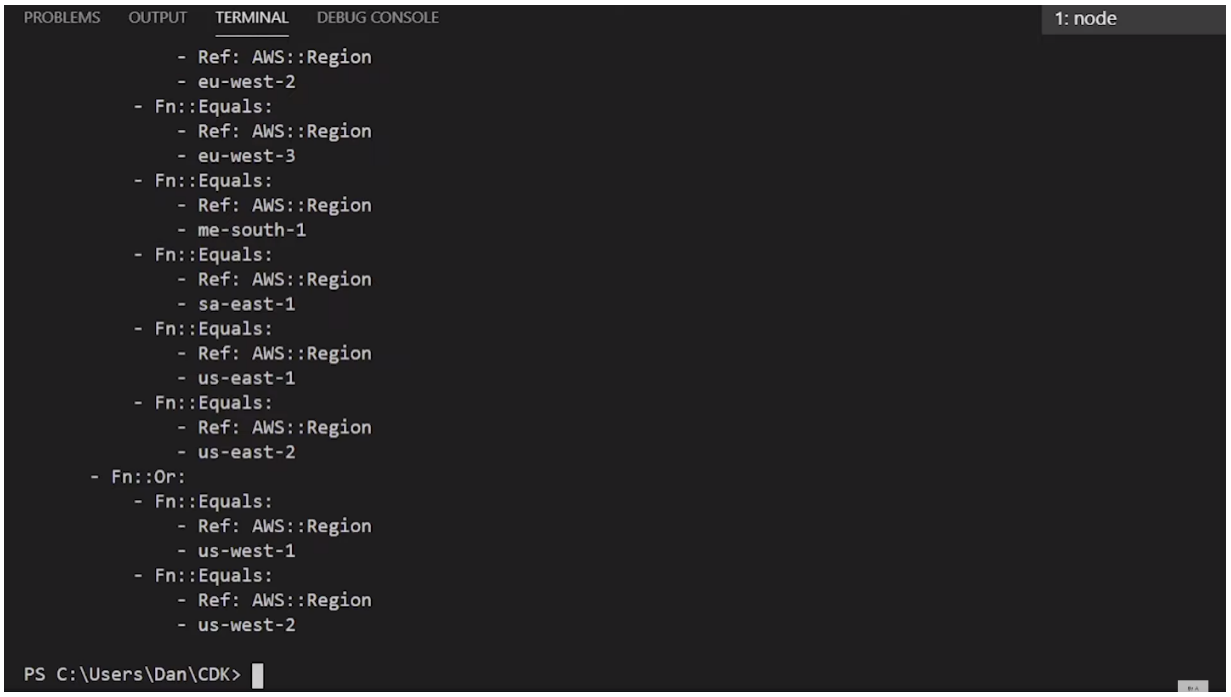


```
PS C:\Users\Dan\CDK> cdk synth
```

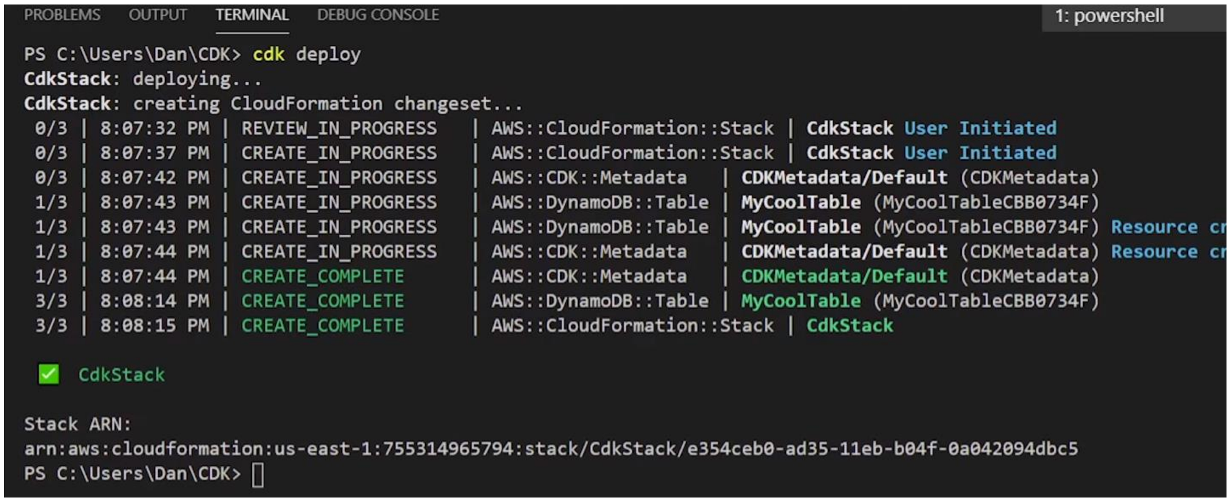
We are now ready for deployment; we will use the **cdk synth** command to synthesize the code into CloudFormation



```
Resources:
  MyCoolTableCBB0734F:
    Type: AWS::DynamoDB::Table
    Properties:
      KeySchema:
        - AttributeName: userId
          KeyType: HASH
        - AttributeName: noteId
          KeyType: RANGE
      AttributeDefinitions:
        - AttributeName: userId
          AttributeType: S
        - AttributeName: noteId
          AttributeType: S
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 5
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
    Metadata:
      aws:cdk:path: CdkStack/MyCoolTable/Resource
  CDKMetadata:
    Type: AWS::CDK::Metadata
    Properties:
      Analytics: v2:deflate64:H4sIAAAAAAAACkWNwQ6DMAxDv2X3ECinHSfxB2w/UNIiFUYitSnTVPXfx9hhJ1t+lm3QdD12l5t9pYbc
rDzP905gO4oEG4wnfBvdlu4iYsDzs9z8ZpagUW53FJ7W56NNfjckkhNDGzhs3j+NMPQWHF448AAAA=
```

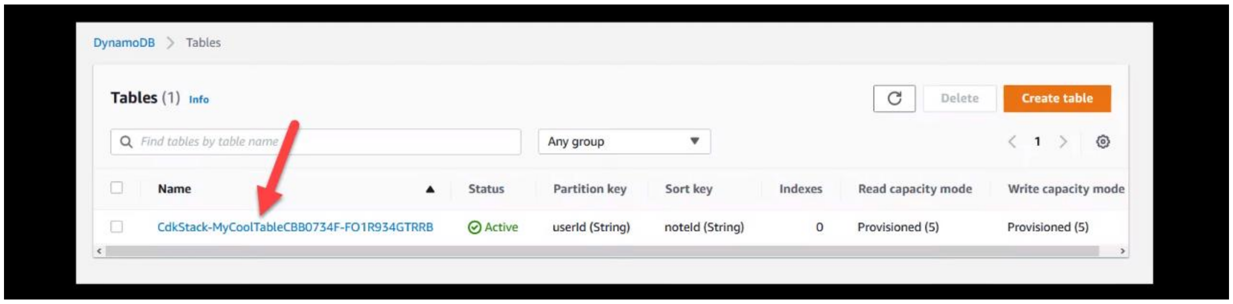


```
            - Ref: AWS::Region
            - eu-west-2
        - Fn::Equals:
            - Ref: AWS::Region
            - eu-west-3
        - Fn::Equals:
            - Ref: AWS::Region
            - me-south-1
        - Fn::Equals:
            - Ref: AWS::Region
            - sa-east-1
        - Fn::Equals:
            - Ref: AWS::Region
            - us-east-1
        - Fn::Equals:
            - Ref: AWS::Region
            - us-east-2
    - Fn::Or:
        - Fn::Equals:
            - Ref: AWS::Region
            - us-west-1
        - Fn::Equals:
            - Ref: AWS::Region
            - us-west-2

PS C:\Users\Dan\CDK>
```



```
PS C:\Users\Dan\CDK> cdk deploy
CdkStack: deploying...
CdkStack: creating CloudFormation changeset...
 0/3 | 8:07:32 PM | REVIEW_IN_PROGRESS   | AWS::CloudFormation::Stack | CdkStack User Initiated
 0/3 | 8:07:37 PM | CREATE_IN_PROGRESS   | AWS::CloudFormation::Stack | CdkStack User Initiated
 0/3 | 8:07:42 PM | CREATE_IN_PROGRESS   | AWS::CDK::Metadata     | CDKMetadata/Default (CDKMetadata)
 1/3 | 8:07:43 PM | CREATE_IN_PROGRESS   | AWS::DynamoDB::Table | MyCoolTable (MyCoolTableCBB0734F)
 1/3 | 8:07:43 PM | CREATE_IN_PROGRESS   | AWS::DynamoDB::Table | MyCoolTable (MyCoolTableCBB0734F) Resource cr
 1/3 | 8:07:44 PM | CREATE_IN_PROGRESS   | AWS::CDK::Metadata     | CDKMetadata/Default (CDKMetadata) Resource cr
 1/3 | 8:07:44 PM | CREATE_COMPLETE      | AWS::CDK::Metadata     | CDKMetadata/Default (CDKMetadata)
 3/3 | 8:08:14 PM | CREATE_COMPLETE      | AWS::DynamoDB::Table | MyCoolTable (MyCoolTableCBB0734F)
 3/3 | 8:08:15 PM | CREATE_COMPLETE      | AWS::CloudFormation::Stack | CdkStack

 ✅  CdkStack

Stack ARN:
arn:aws:cloudformation:us-east-1:755314965794:stack/CdkStack/e354ceb0-ad35-11eb-b04f-0a042094dbc5
PS C:\Users\Dan\CDK>
```

We can now deploy this to AWS, the CLI provides use with updates as the resources are created



Name	Status	Partition key	Sort key	Indexes	Read capacity mode	Write capacity mode
CdkStack-MyCoolTableCBB0734F-FO1R934GTRRB	Active	userId (String)	noteId (String)	0	Provisioned (5)	Provisioned (5)

We then see our table.