**DynamoDB:** Used for Data Replication

=========================================================

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. ... With DynamoDB, you can create database tables that can store and retrieve any amount of data and serve any level of request traffic.

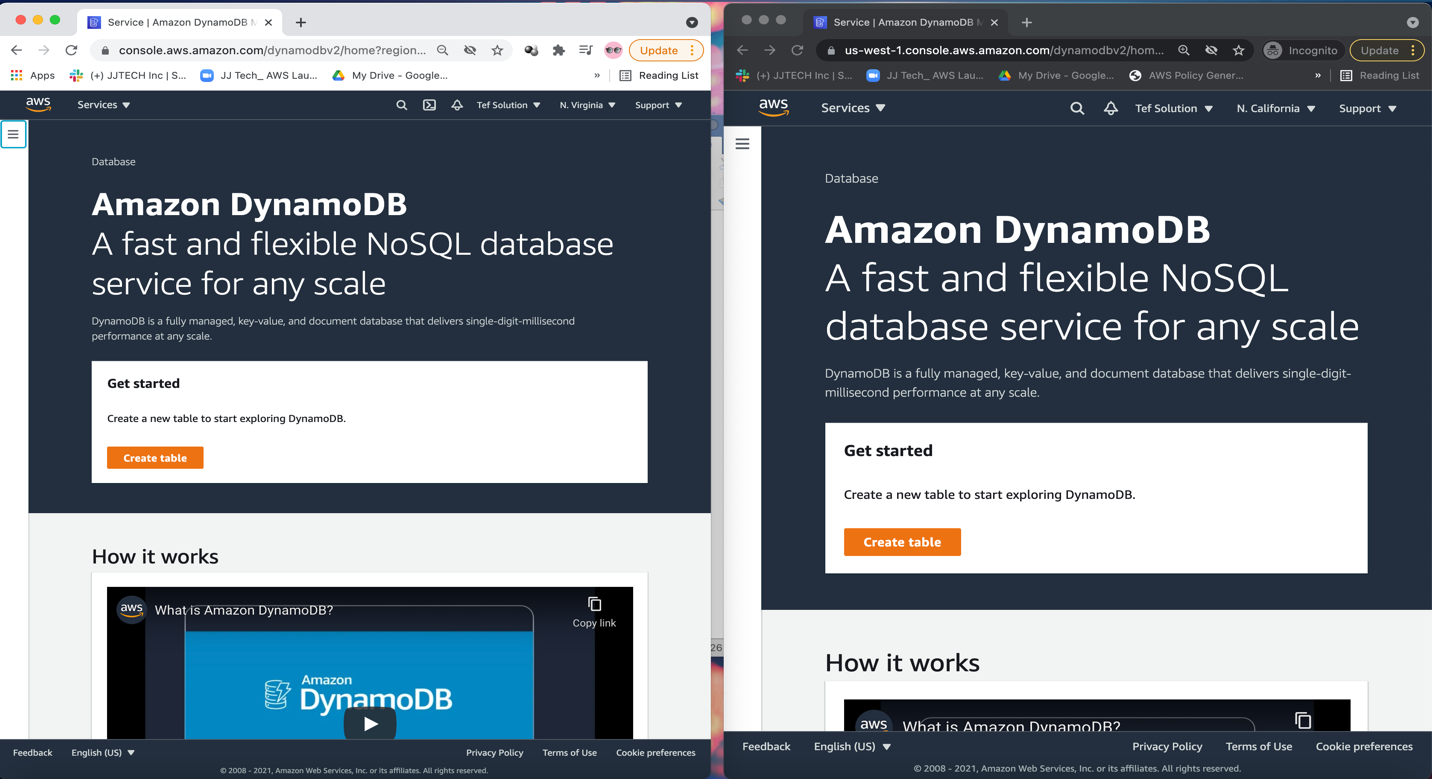
* When an application writes data to a replica table in one Region, DynamoDB propagates the write to the other replica tables in the other AWS Regions automatically.

**Use-case:** Create a table in N. Virginia and have its data automatically replicated to N. California.

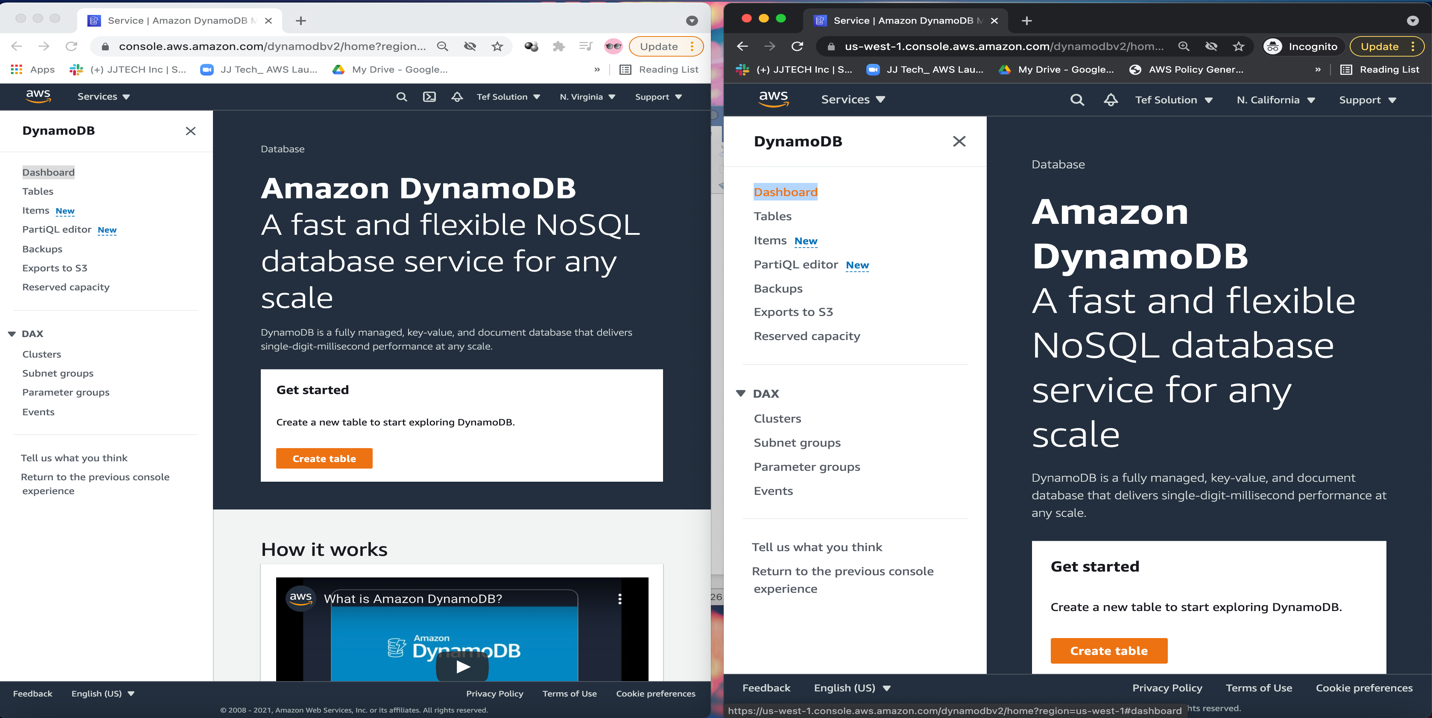
**Create a DynamoDB Global tables:**

🡪 Open your AWS account on two separate browsers and search for DynamoDB.

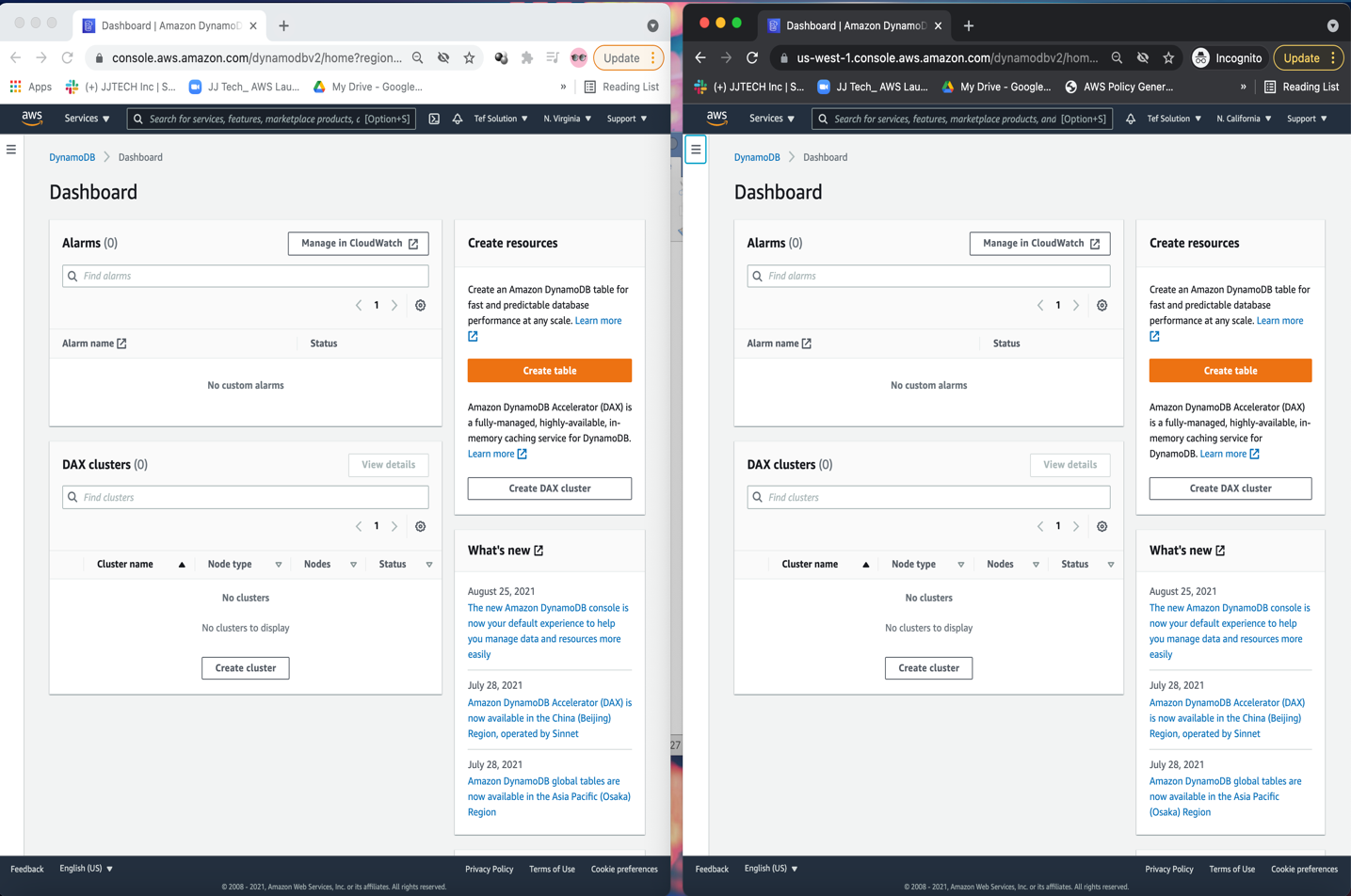
* One browser should be in N. Virginia and the other in N. California.



🡪 On both accounts, click on Dashboard.

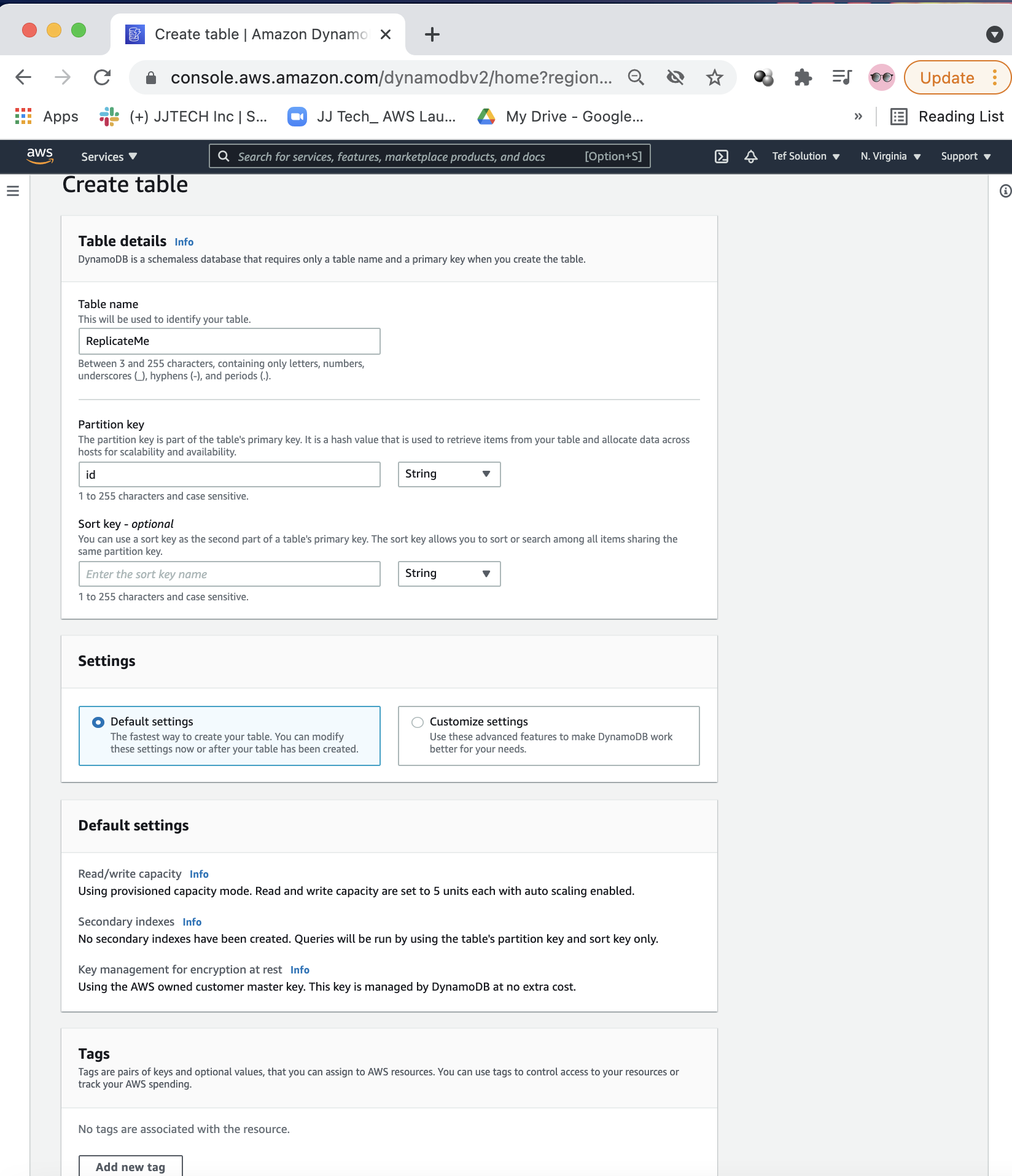


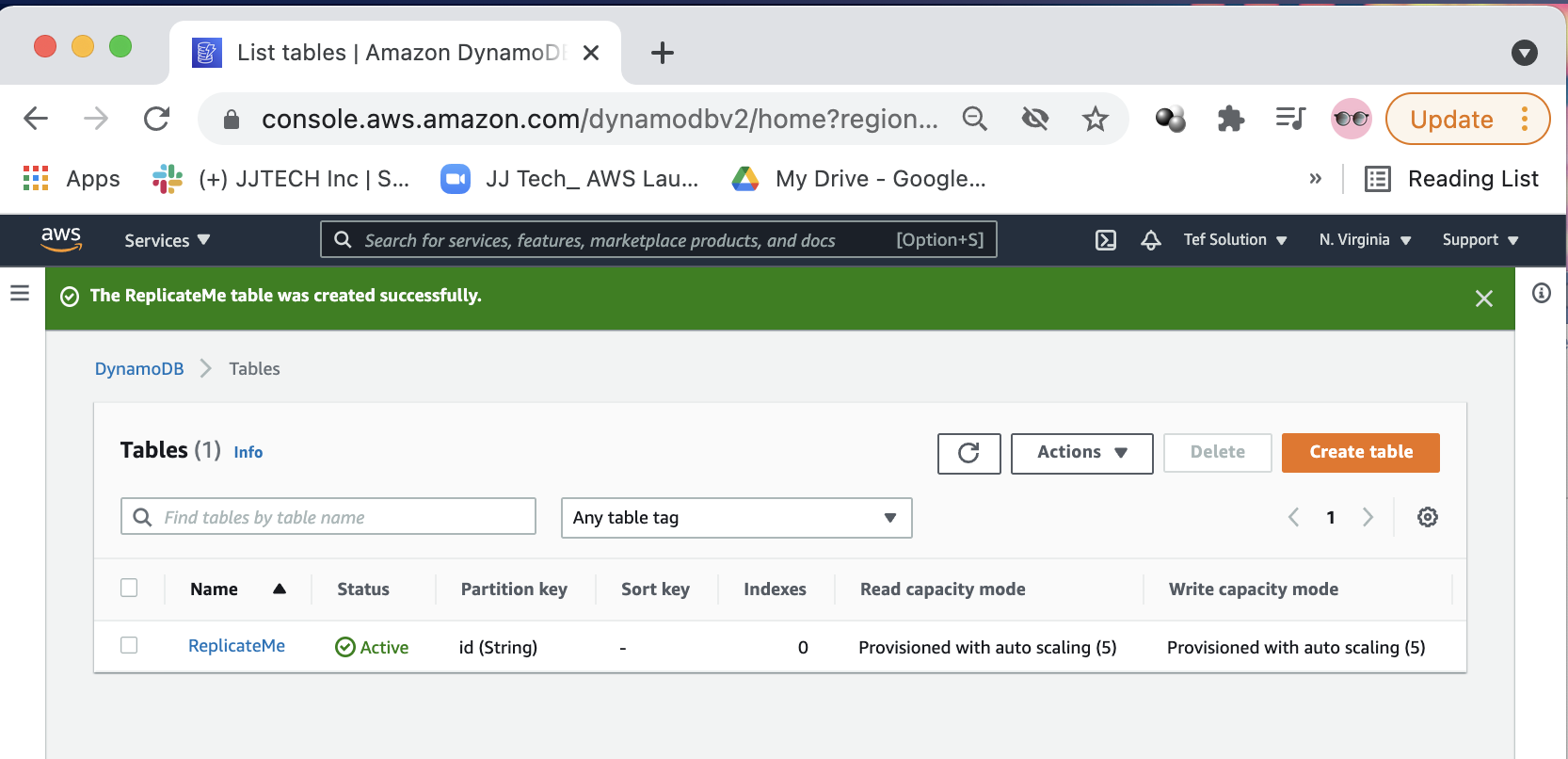
🡪 Create a table in N. Virginia and have its data automatically replicated to N. California.



In **NOVA**, click on create table.

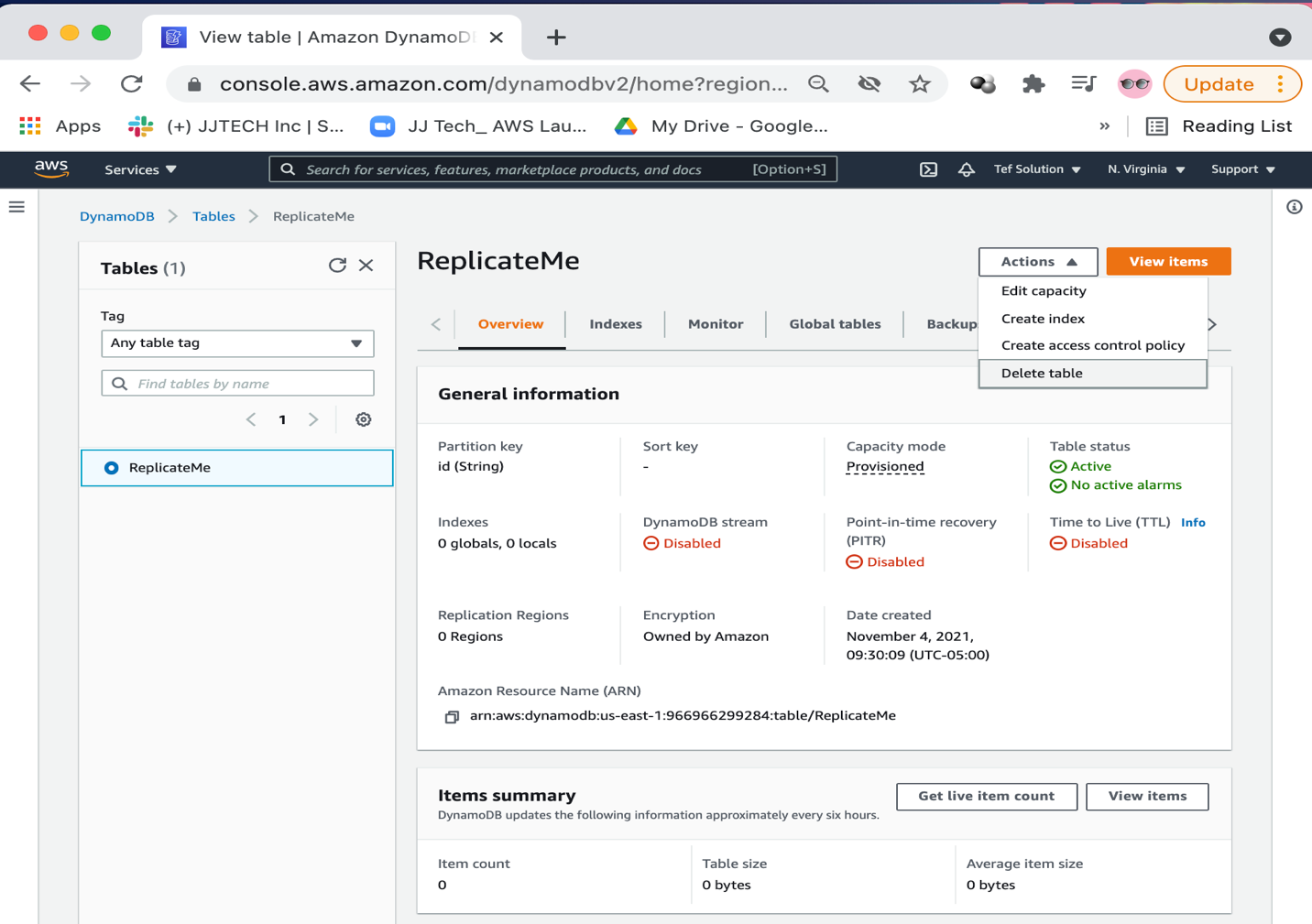
* Name the table: ReplicateMe
* Partition Key: id
* Click on **create**



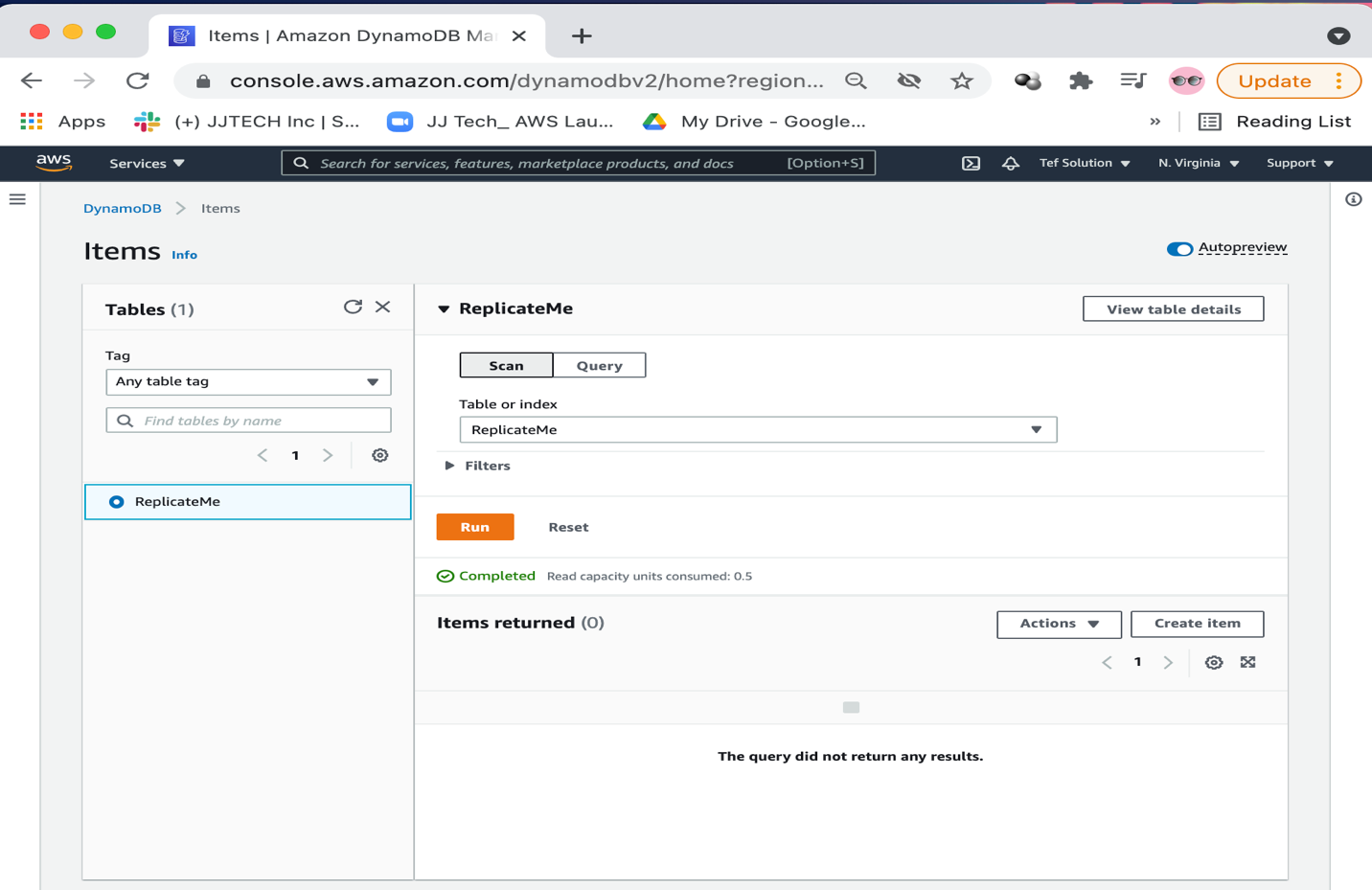


🡪 Click on the table.

🡪 Click on view items



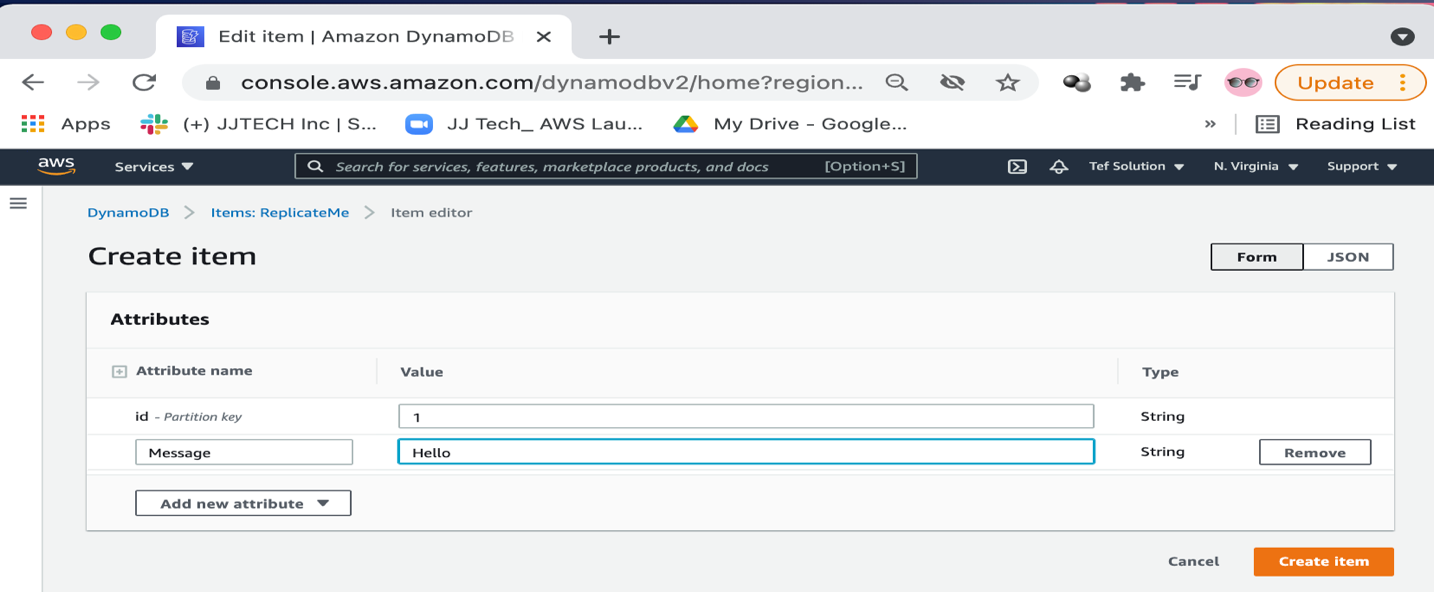
🡪 Click on create item.



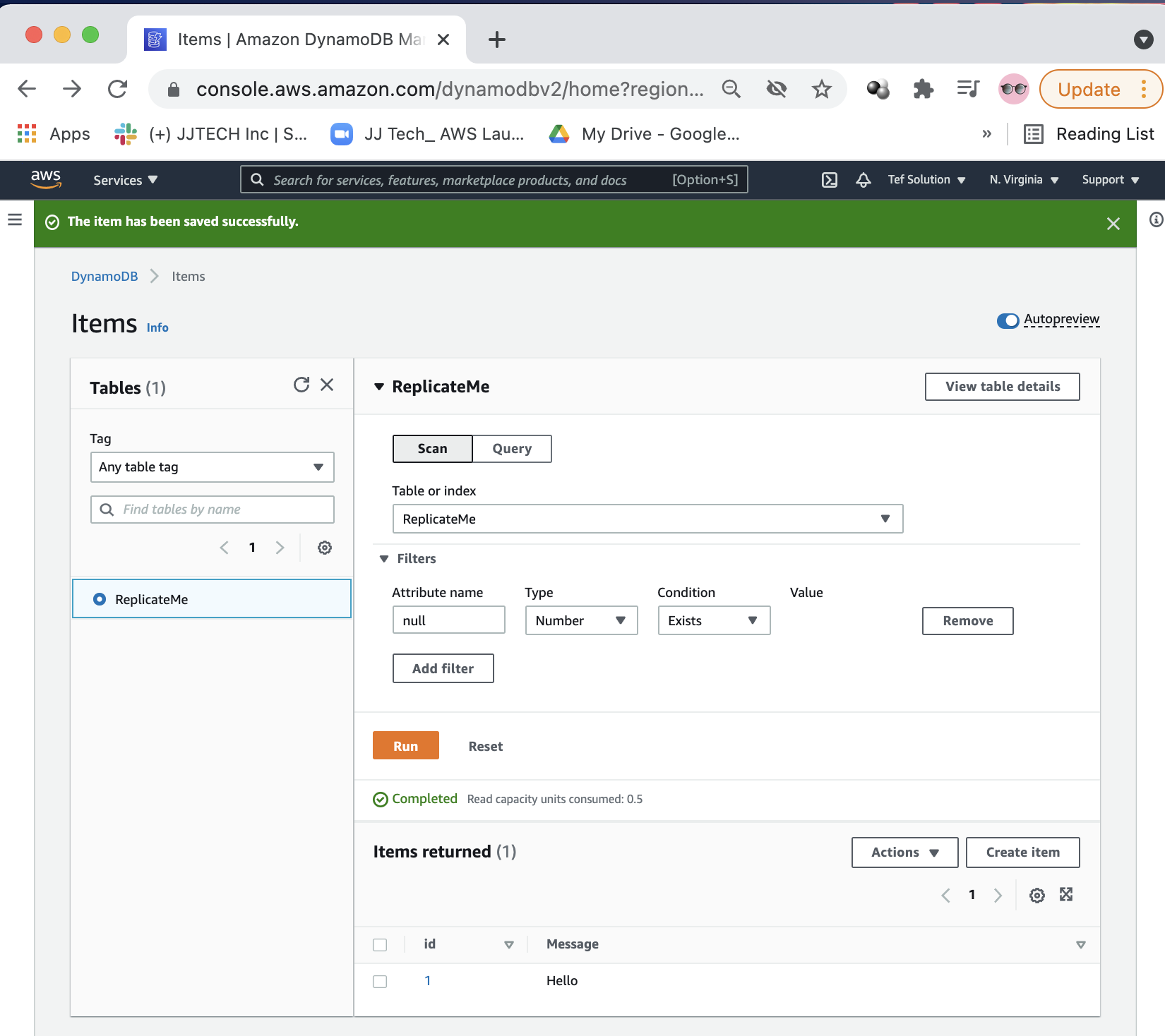
Before we start replicating data, let’s create an item in the source table (add Data into that table).

🡪 Click on create item again.

* Insert: 1 for value - Partition key
* Add new attribute and name it: Message and Value: Hello

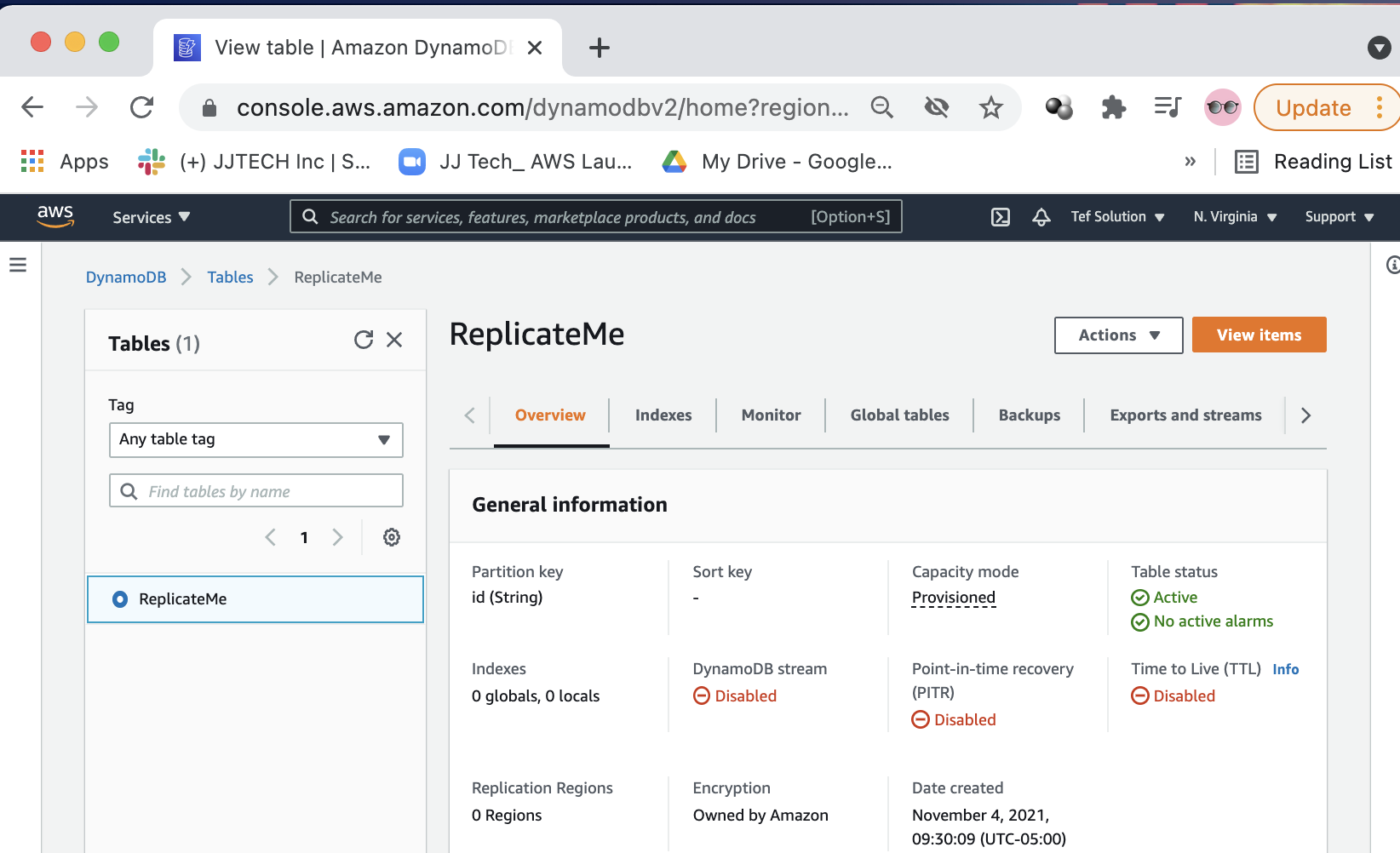


🡪 Click create item.

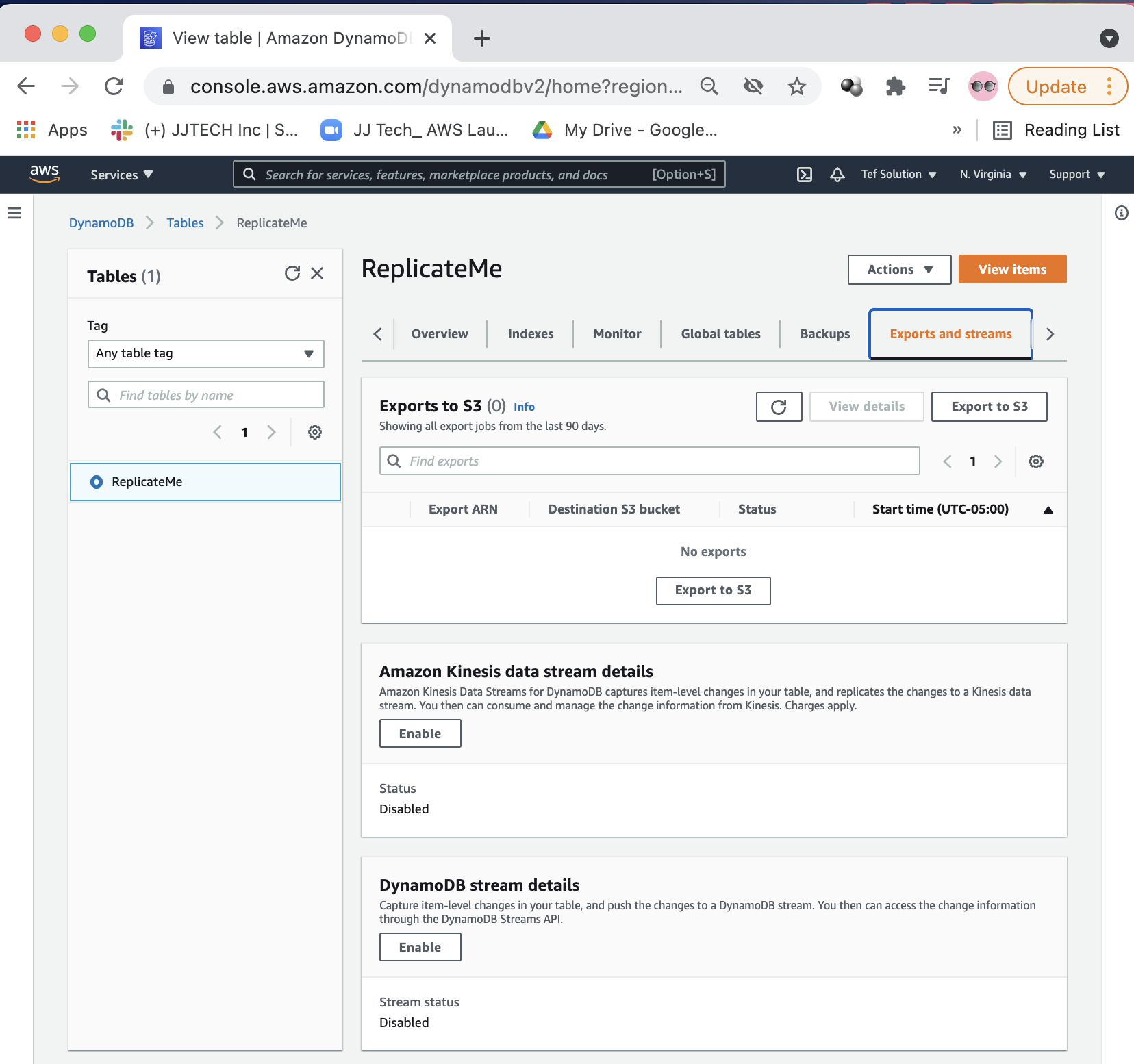


**Setup Global tables to start replicating data from NOVA to N. California.**

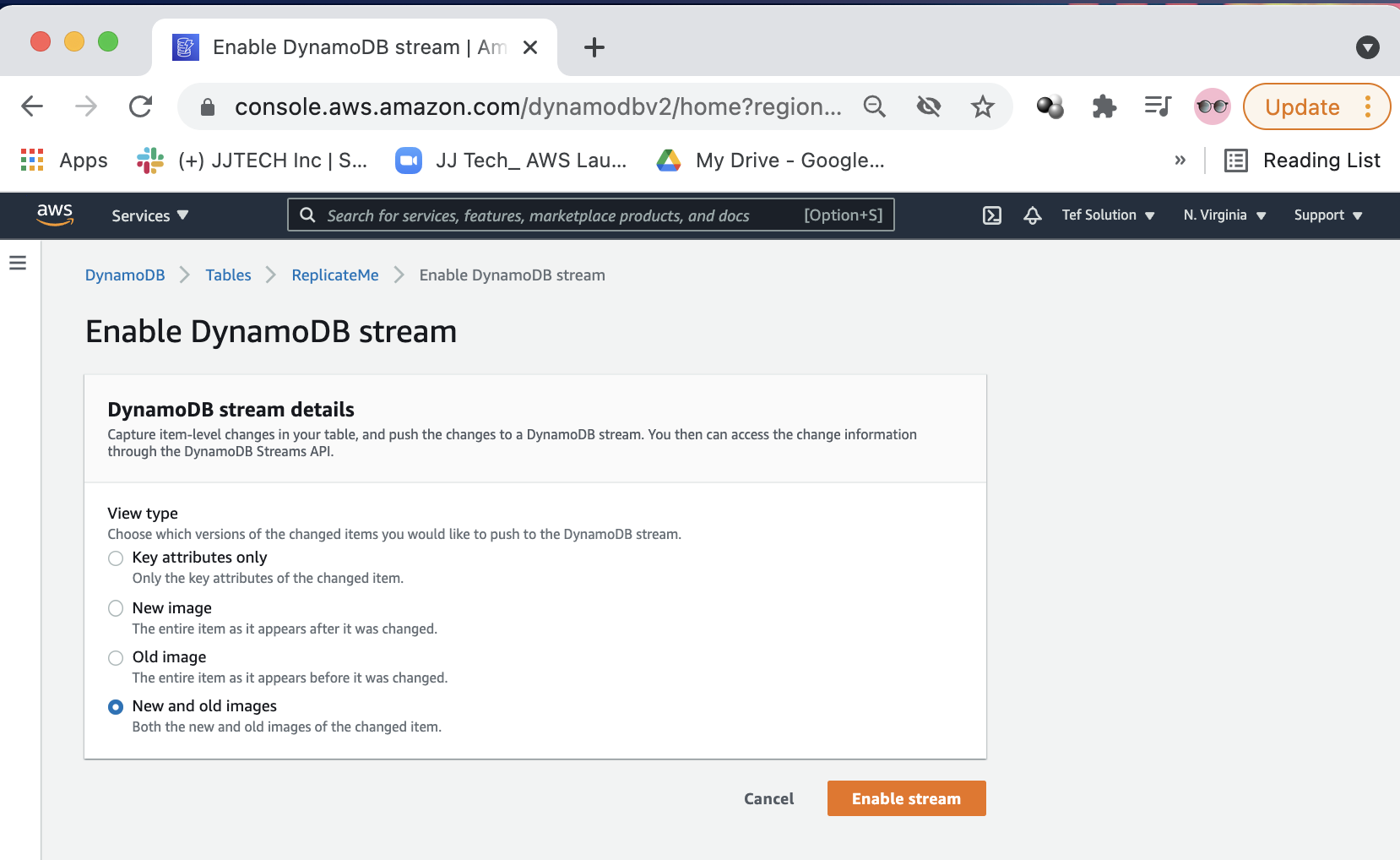
🡪 Click on the ReplicateMe table 🡪 choose Global tables.



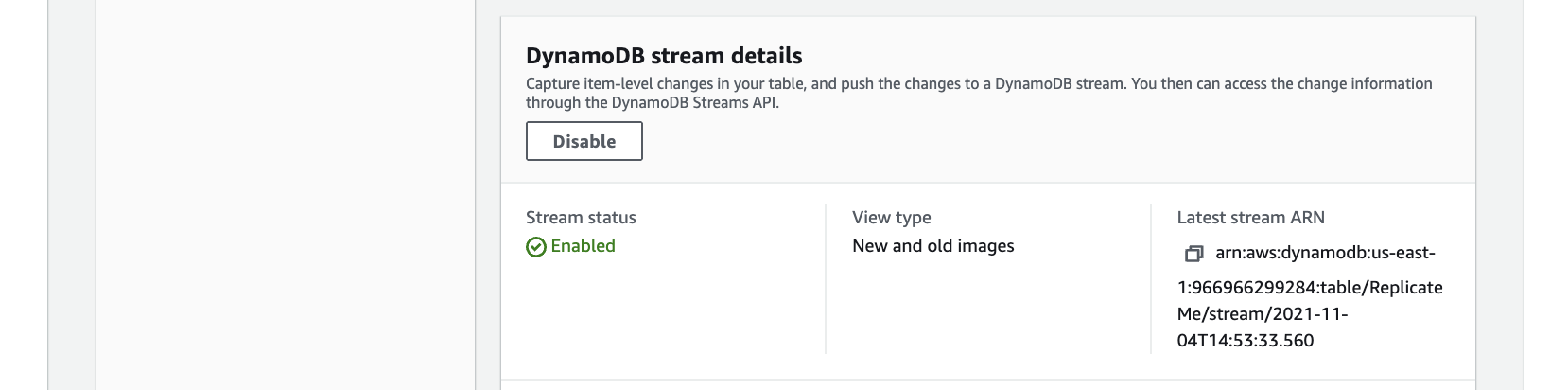
🡪 Click on **Exports and Streams** and enable DynamoDB stream.



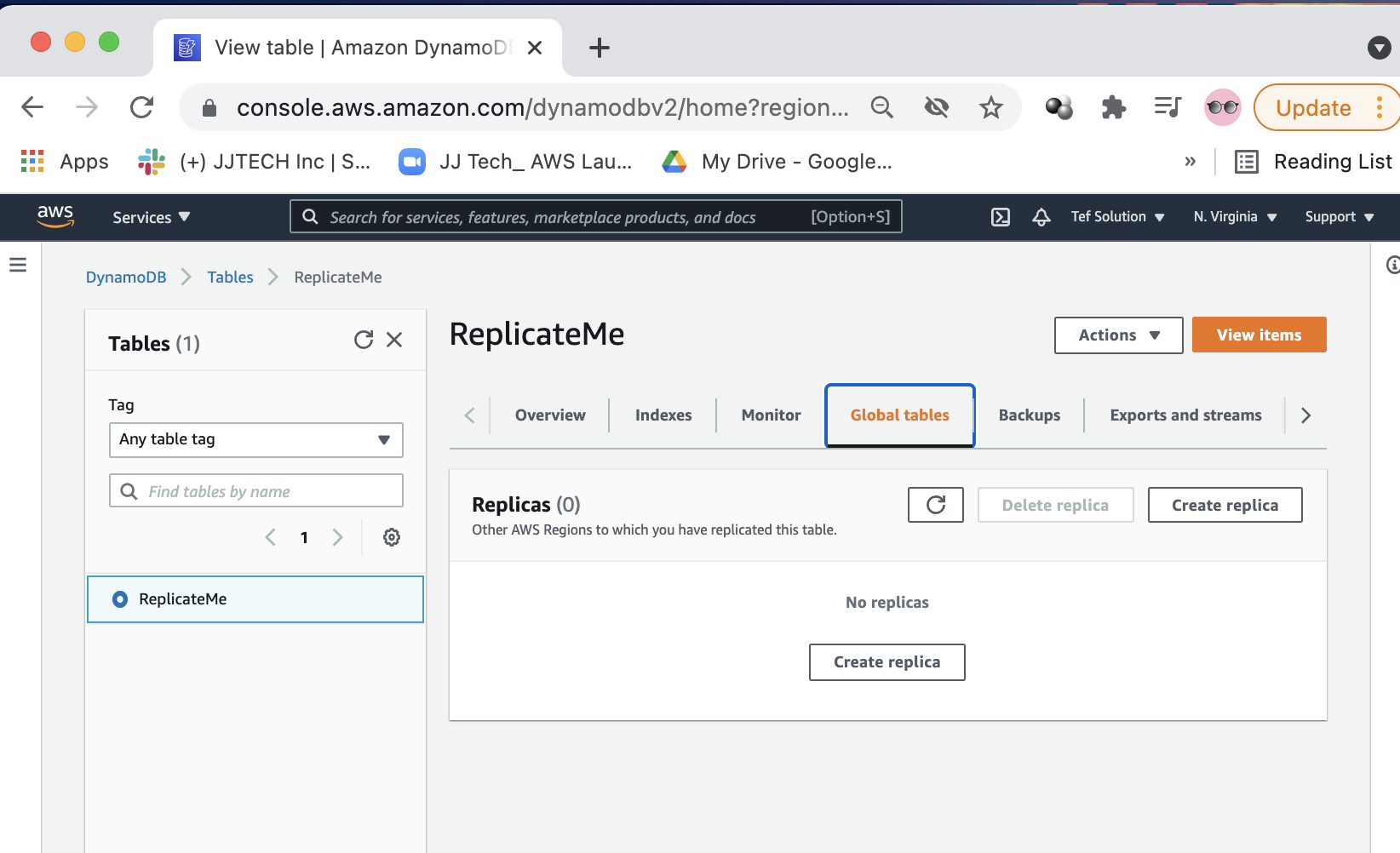
🡪 Choose: New and old images!!



🡪 Click **enable stream**.

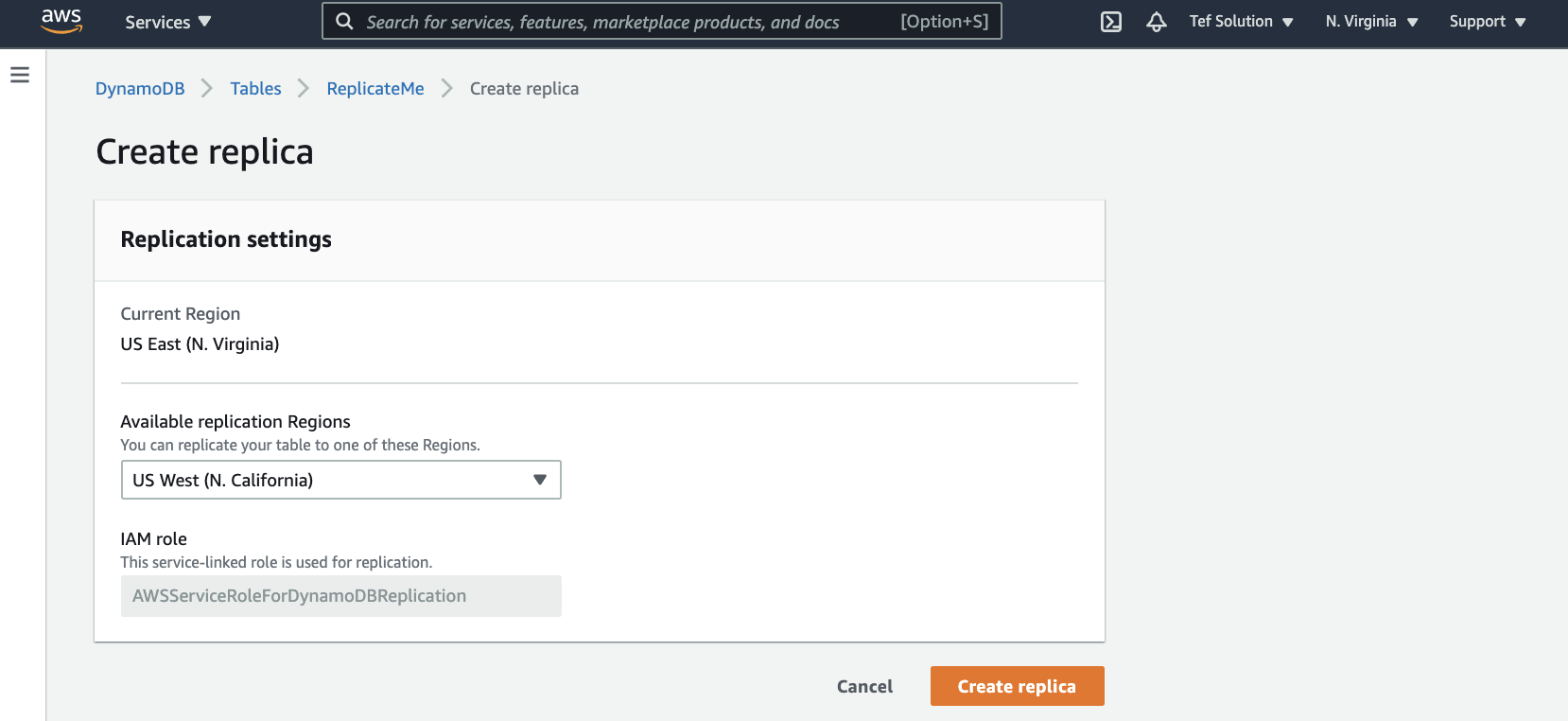


🡪 Click on Global tables.



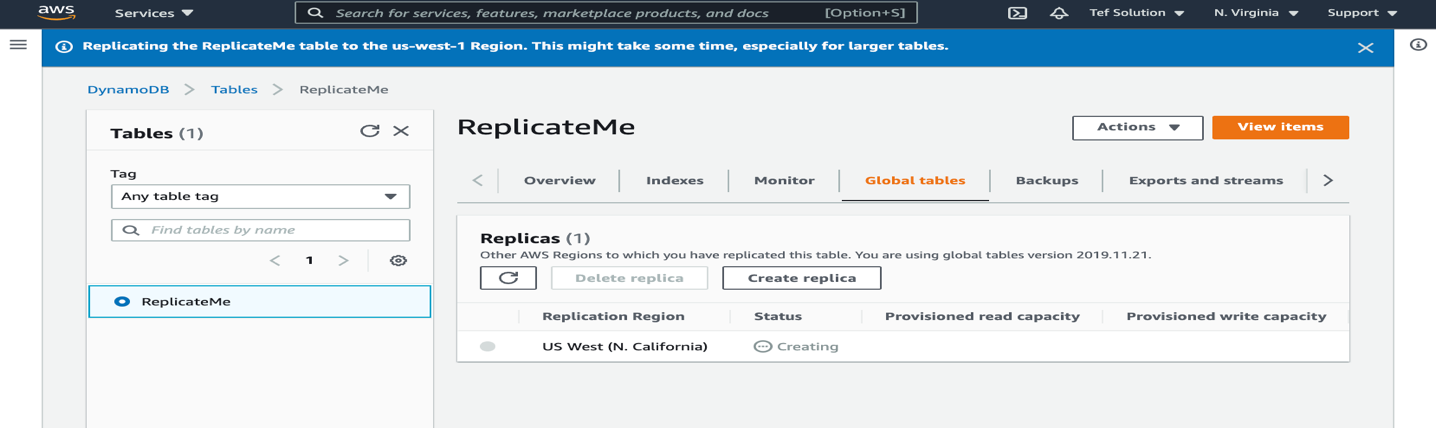
🡪 Click on create replica.

* Choose region: N. California which is the region where you are logged into the other account.

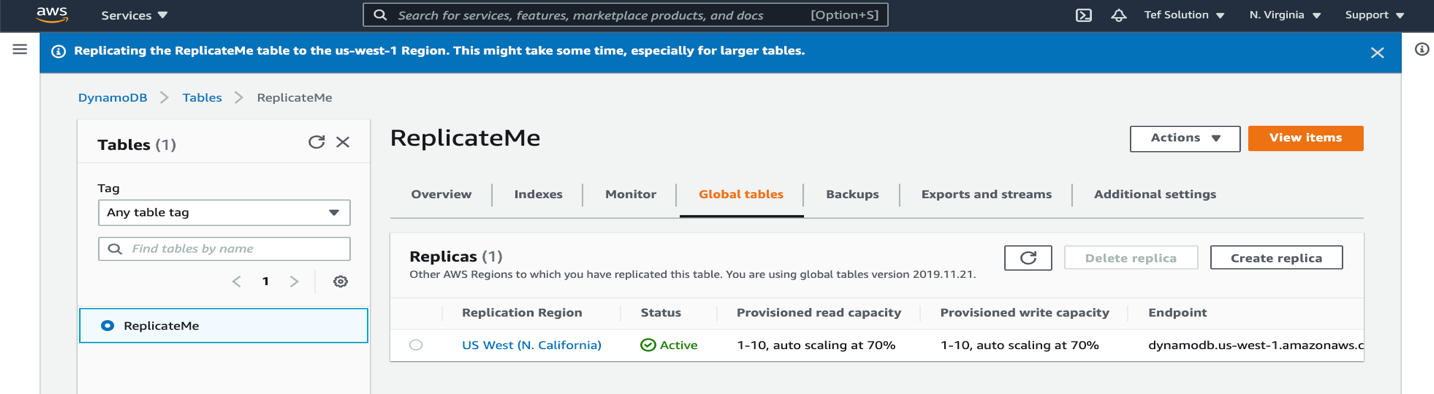


🡪 Click on create replica.

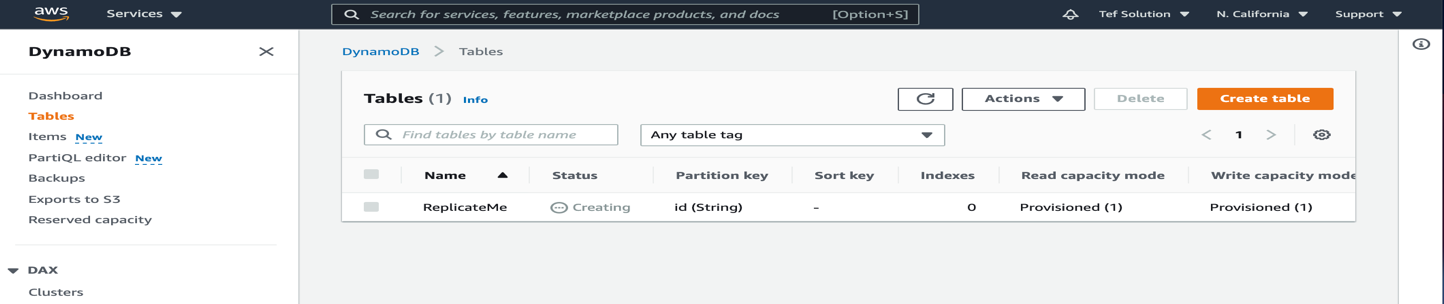
Replication is creating!!



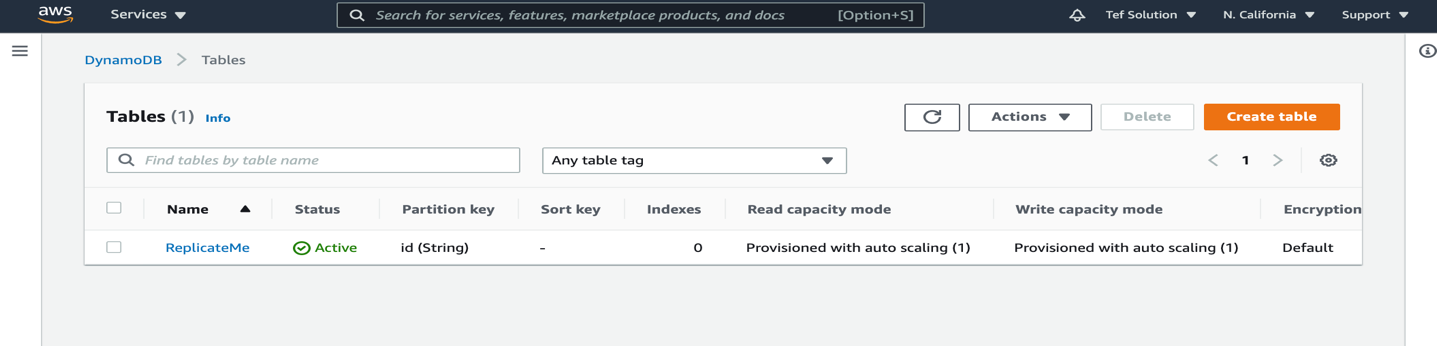
Replication is now active:



In your N. California account, refresh it and you will see the new table appear there.



It is now active in N. California:



**TESTING:**

After it done creating, you can Test it again by creating a new item on the ReplicateME table in NOVA and see how fast it gets replicated to N. California.

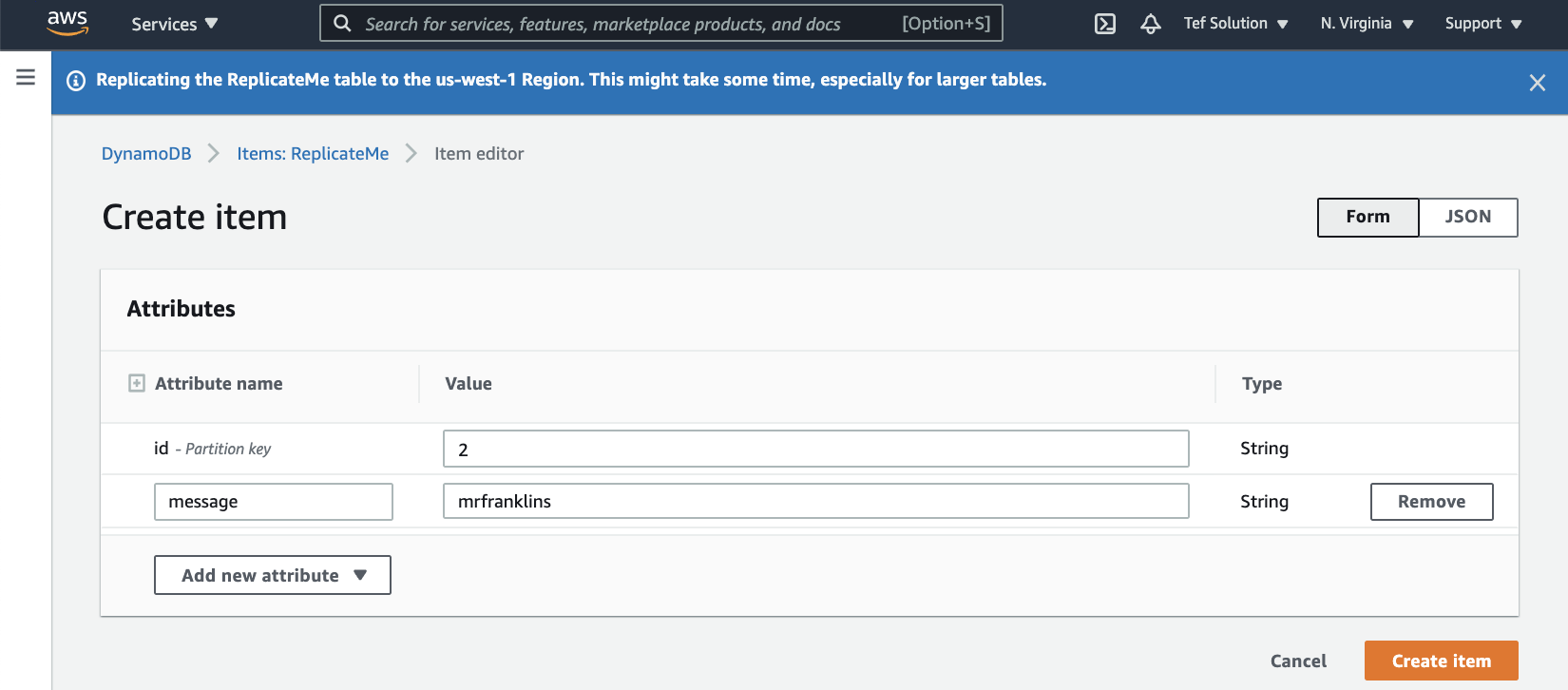
In Nova, create a new item.

🡪 Click on the ReplicateMe table.

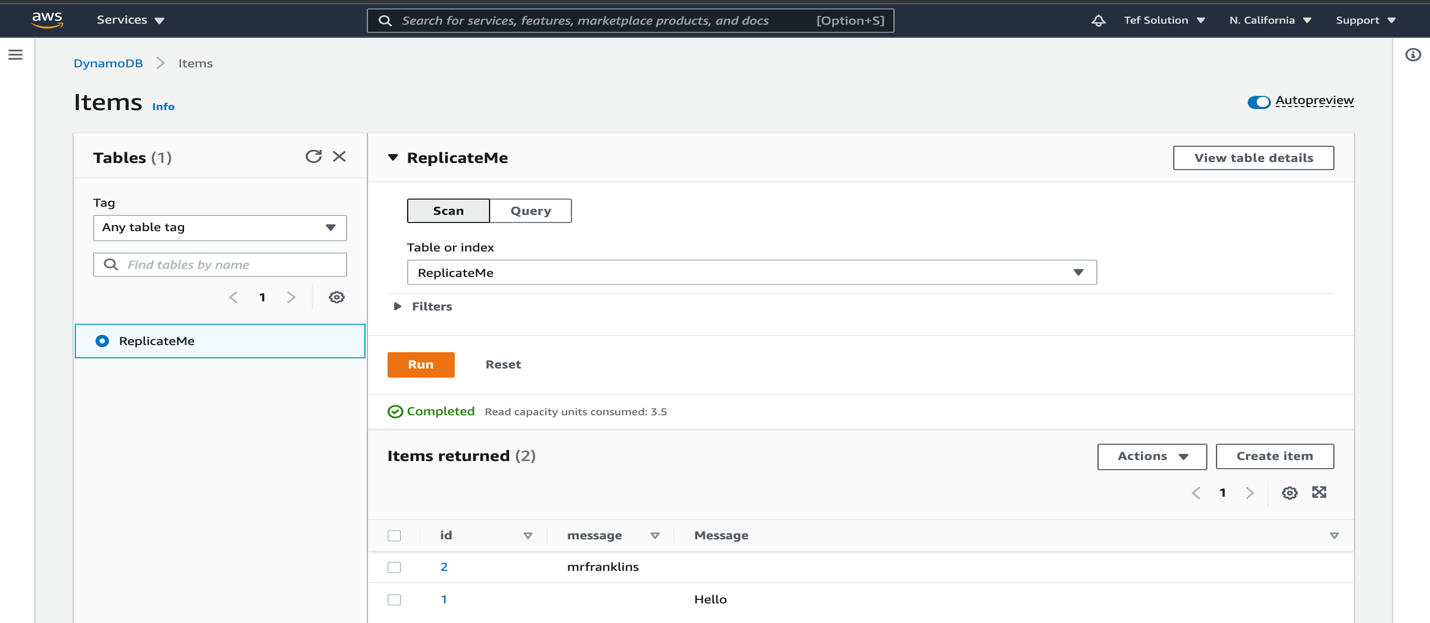
🡪 Click on View items.

🡪 Click on create item.

* Insert: 2 for value - Partition key
* Add new attribute and name it: Message and Value: mrfranklins



Now, go to N. California. In the ReplicateMe table, under items. You will instantly see the new item you added in NOVA.



DONE!!!

### **CONGRATULATIONS!!**

### **YOU’VE SUCCESSFULLY CREATED AND TESTED DYNAMODB TABLES**

Cleanup….

🡪 Delete the ReplicateMe table from N. California first

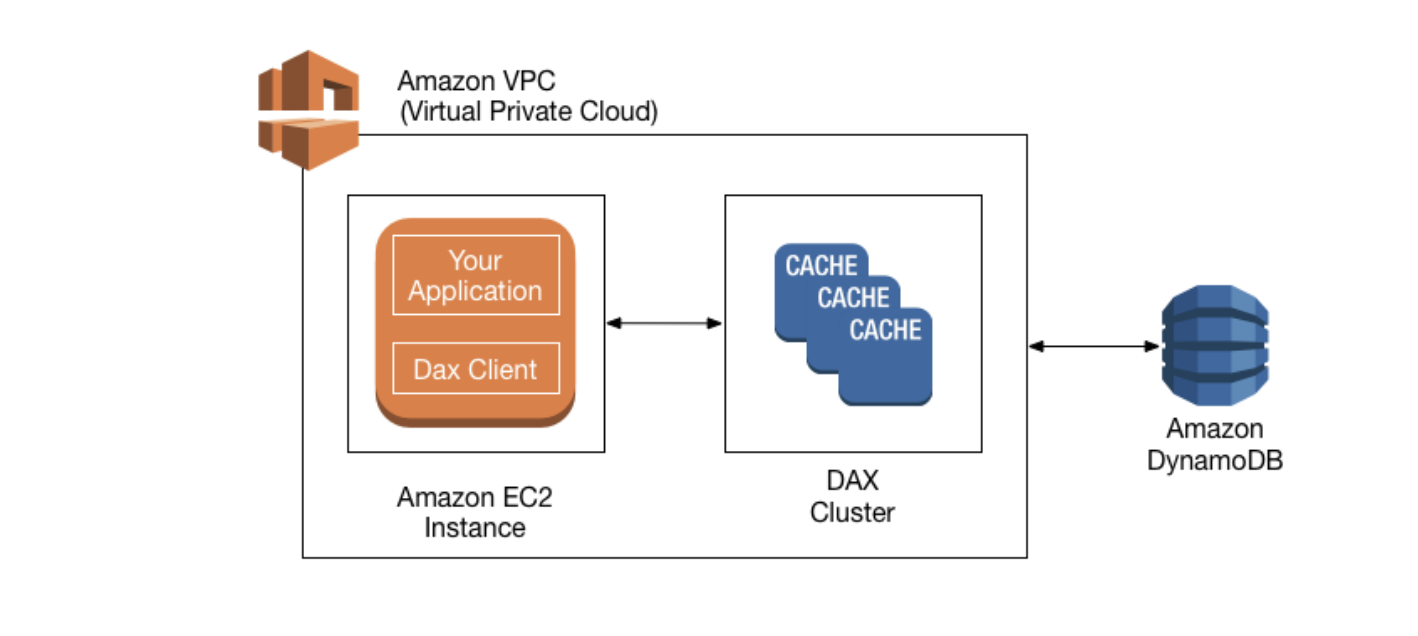
🡪 In NOVA, delete the 2 items create then delete the table.

* You have check both boxes of create backup copy before deleting and acknowledgement. If not, it will not delete.

**EXTRA NOTES:**

**Amazon DynamoDB Accelerator (DAX):** Caching for DynamoDB

* Amazon DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory cache for Amazon DynamoDB that delivers up to a 10 times performance improvement—from milliseconds to microseconds—even at millions of requests per second.



How DAX Processes Requests

A DAX cluster consists of one or more nodes. Each node runs its own instance of the DAX caching software. One of the nodes serves as the primary node for the cluster. Additional nodes (if present) serve as read replicas. For more information, see Nodes.

Your application can access DAX by specifying the endpoint for the DAX cluster. The DAX client software works with the cluster endpoint to perform intelligent load balancing and routing.

**Read Operations**

DAX can respond to the following API calls:

* GetItem
* BatchGetItem
* Query
* Scan

If the request specifies *eventually consistent reads* (the default behavior), it tries to read the item from DAX:

* If DAX has the item available (a *cache hit*), DAX returns the item to the application without accessing DynamoDB.
* If DAX does not have the item available (a *cache miss*), DAX passes the request through to DynamoDB. When it receives the response from DynamoDB, DAX returns the results to the application. But it also writes the results to the cache on the primary node.

**Note**

If there are any read replicas in the cluster, DAX automatically keeps the replicas in sync with the primary node. For more information, see Clusters.

If the request specifies *strongly consistent reads*, DAX passes the request through to DynamoDB. The results from DynamoDB are not cached in DAX. Instead, they are simply returned to the application.

**Write Operations**

The following DAX API operations are considered "write-through":

* BatchWriteItem
* UpdateItem
* DeleteItem
* PutItem

With these operations, data is first written to the DynamoDB table, and then to the DAX cluster. The operation is successful only if the data is successfully written to *both* the table and to DAX.

**Other Operations**

DAX does not recognize any DynamoDB operations for managing tables (such as CreateTable, UpdateTable, and so on). If your application needs to perform these operations, it must access DynamoDB directly rather than using DAX.

For detailed information about DAX and DynamoDB consistency, see DAX and DynamoDB Consistency Models.

For information about how transactions work in DAX, see Using Transactional APIs in DynamoDB Accelerator (DAX).

**Request Rate Limiting**

If the number of requests sent to DAX exceeds the capacity of a node, DAX limits the rate at which it accepts additional requests by returning a ThrottlingException. DAX continuously evaluates your CPU utilization to determine the volume of requests it can process while maintaining a healthy cluster state.

## What are DynamodDB Streams? How do they work?

DynamoDB Stream can be described as a stream of observed changes in data. Once enabled, whenever you perform a write operation to the DynamoDB table, like put, update or delete, a corresponding event containing information like which record was changed and what was changed will be saved to the Stream.

## Characteristics of DynamoDB Stream

* events are stored up to 24 hours
* ordered, sequence of events in the stream reflects the actual sequence of operations in the table
* near-real time, events are available in the stream within less than a second from the moment of the write operation
* deduplicated, each modification corresponds to exactly one record within the stream

