**Real-Time Data Processing with Serverless Architecture on AWS**

**Project Abstract:**

This project focuses on leveraging AWS services to implement a serverless architecture for real-time data processing. The core components include an empty DynamoDB table for data storage and two AWS Lambda functions: a producer lambda responsible for continuous data ingestion from a data pipeline to a Kinesis Data Stream, and a consumer lambda triggered by the Kinesis Data Stream to process and store incoming data into the DynamoDB table. This serverless architecture offers scalability, flexibility, and cost-efficiency, making it suitable for a wide range of real-time data processing applications across various industries.

**Services Used:**

1. DynamoDB: A fully managed NoSQL database service for high-performance, scalable, and low-latency storage of structured data.
2. AWS Lambda: A serverless compute service that allows you to run code in response to events without managing servers, providing automatic scaling and high availability.
3. AWS Kinesis Data Stream: A real-time data streaming service for ingesting, processing, and analyzing large volumes of streaming data in real-time with low latency.

**Project Details Steps (as Completed by Me):**

1. **Create an Empty DynamoDB Table:**
   * Log in to the AWS Management Console.
   * Navigate to the DynamoDB service.
   * Click on "Create table."
   * Enter a name for the table and define the primary key.
   * Choose appropriate settings for read and write capacity.
   * Click on "Create table."

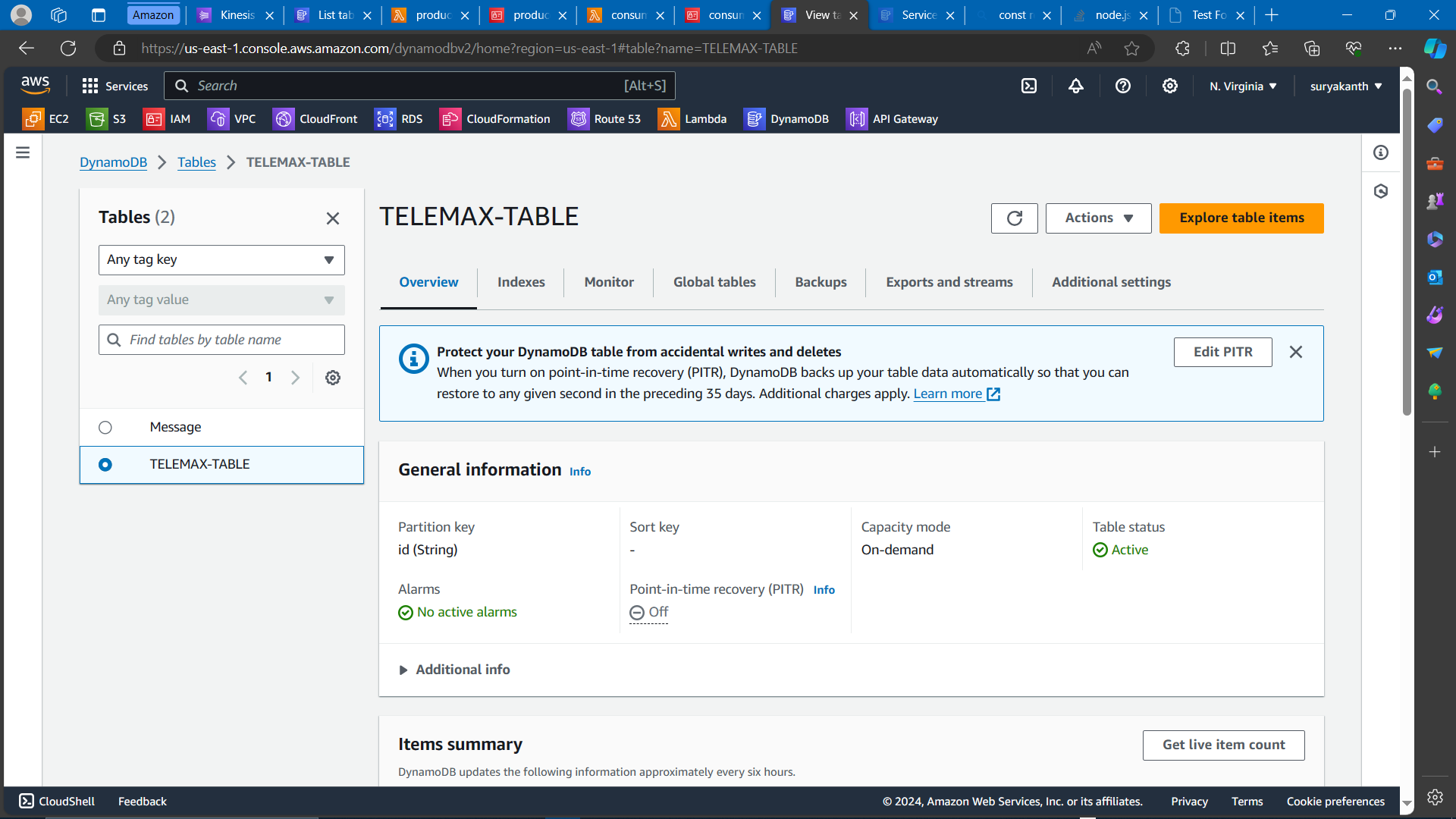


Fig 1 – Empty dynamoDB table created with primary key id

1. **Create Two Lambda Functions:**
   * Navigate to the Lambda service in the AWS Management Console.
   * Click on "Create function."
   * Name the first function as "ProducerLambda" and the second one as "ConsumerLambda."
   * Choose a runtime for both functions (e.g., Node.js, Python).
   * Write the code for each lambda function according to their functionality.
   * For the ProducerLambda, the code should continuously send data to the Kinesis Data Stream.
   * For the ConsumerLambda, the code should process incoming data from the Kinesis Data Stream and store it in DynamoDB.
   * Click on "Deploy" or "Save" to save the Lambda functions.

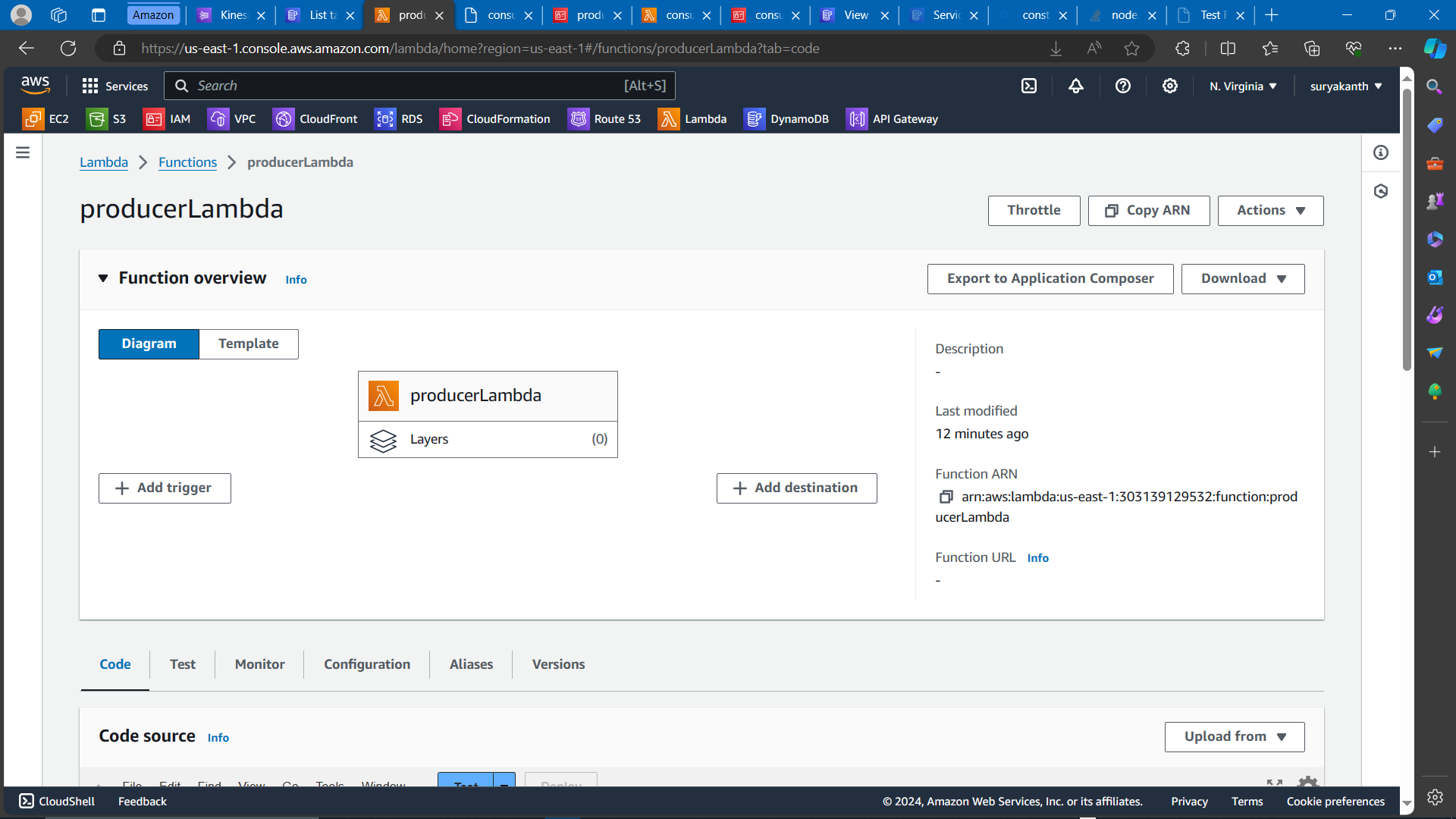
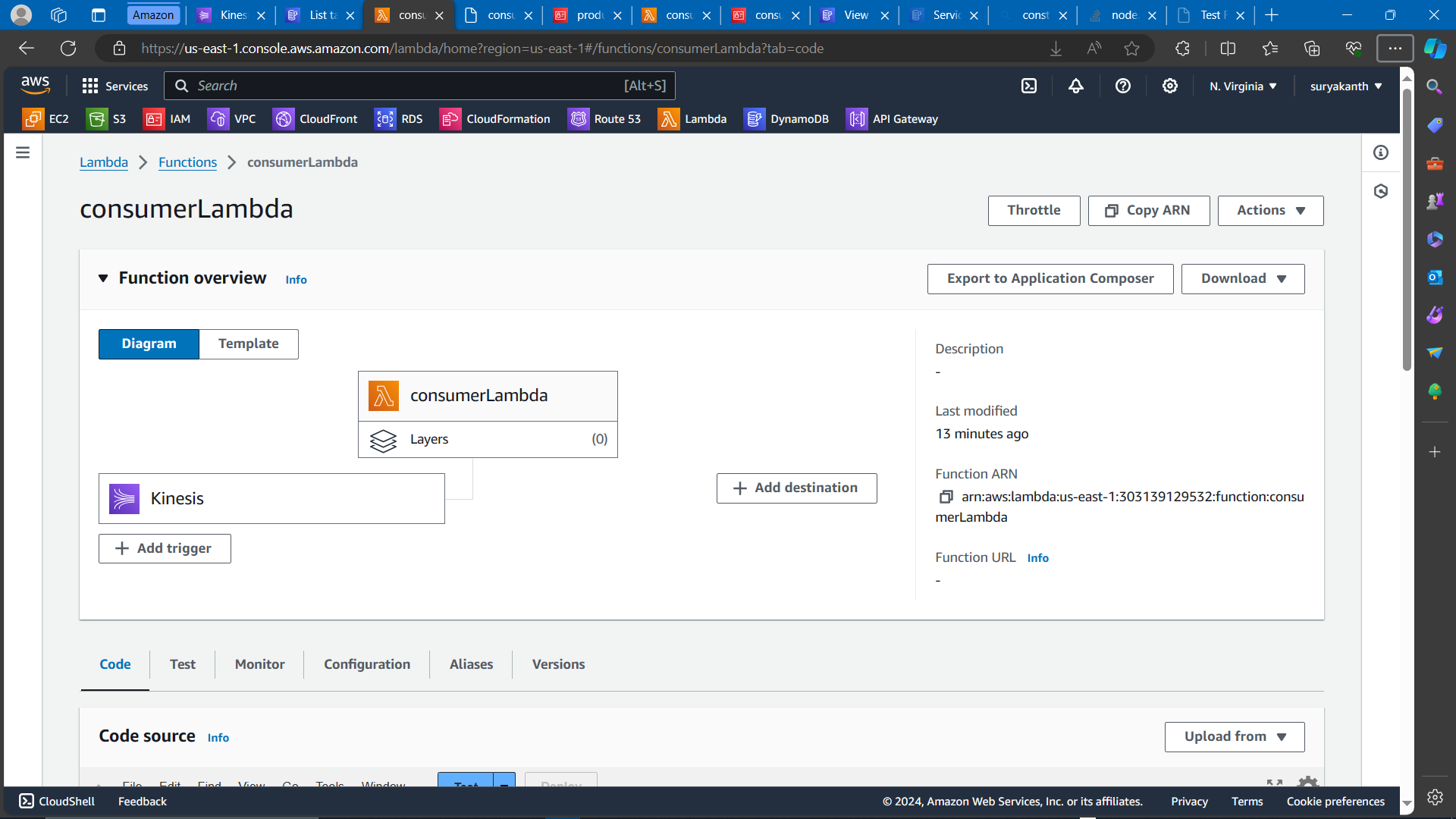
  
  
Fig 2 – Producer lambda function to send data to Kinesis Datastream  
  


Fig 3 – Consumer lambda function to create data in dynamoDB table from Kinesis Datastream

1. **Set up Kinesis Data Stream:**
   * Navigate to the Kinesis service in the AWS Management Console.
   * Click on "Create data stream."
   * Enter a name for the stream and define the number of shards.
   * Click on "Create data stream."

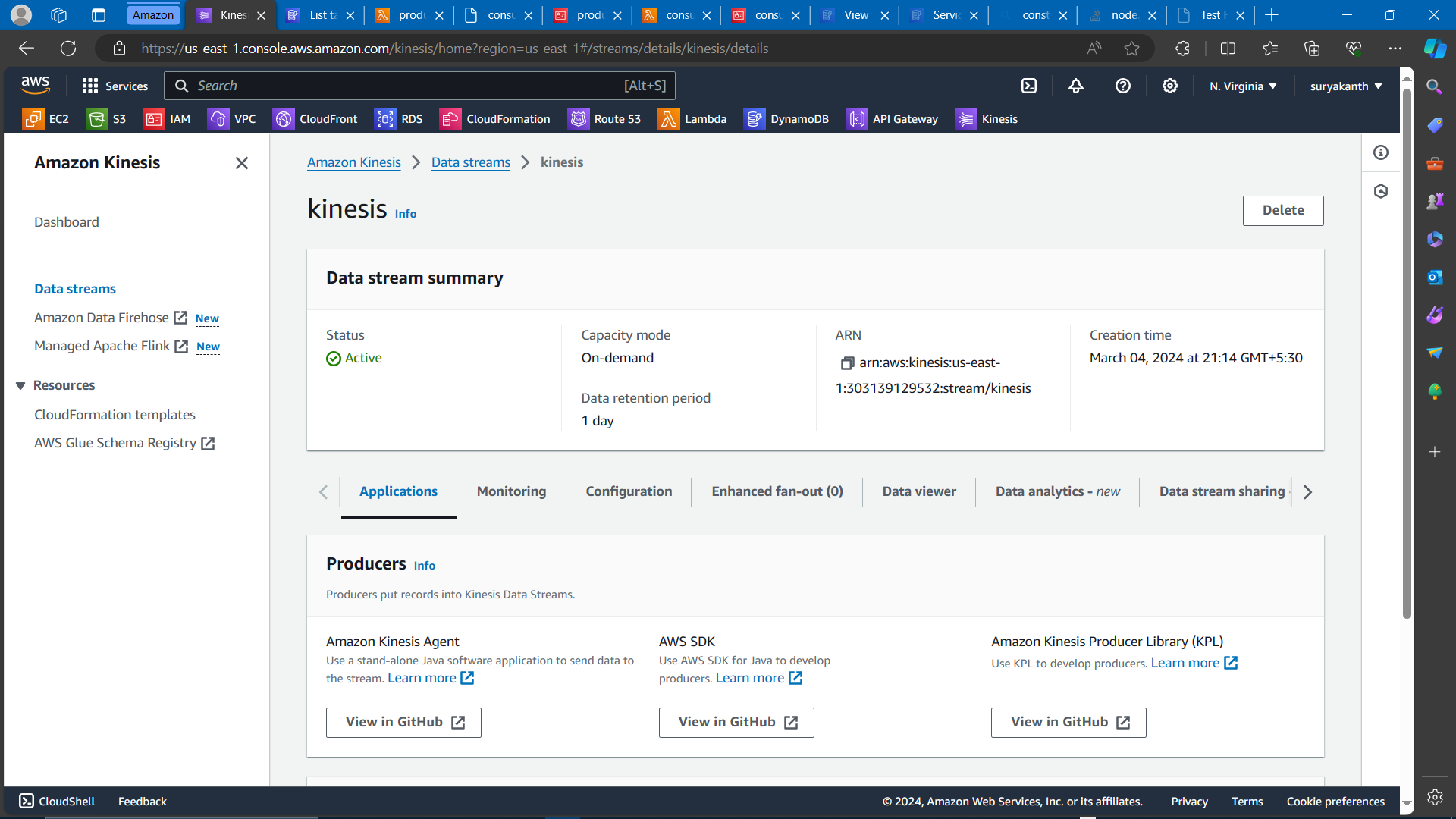
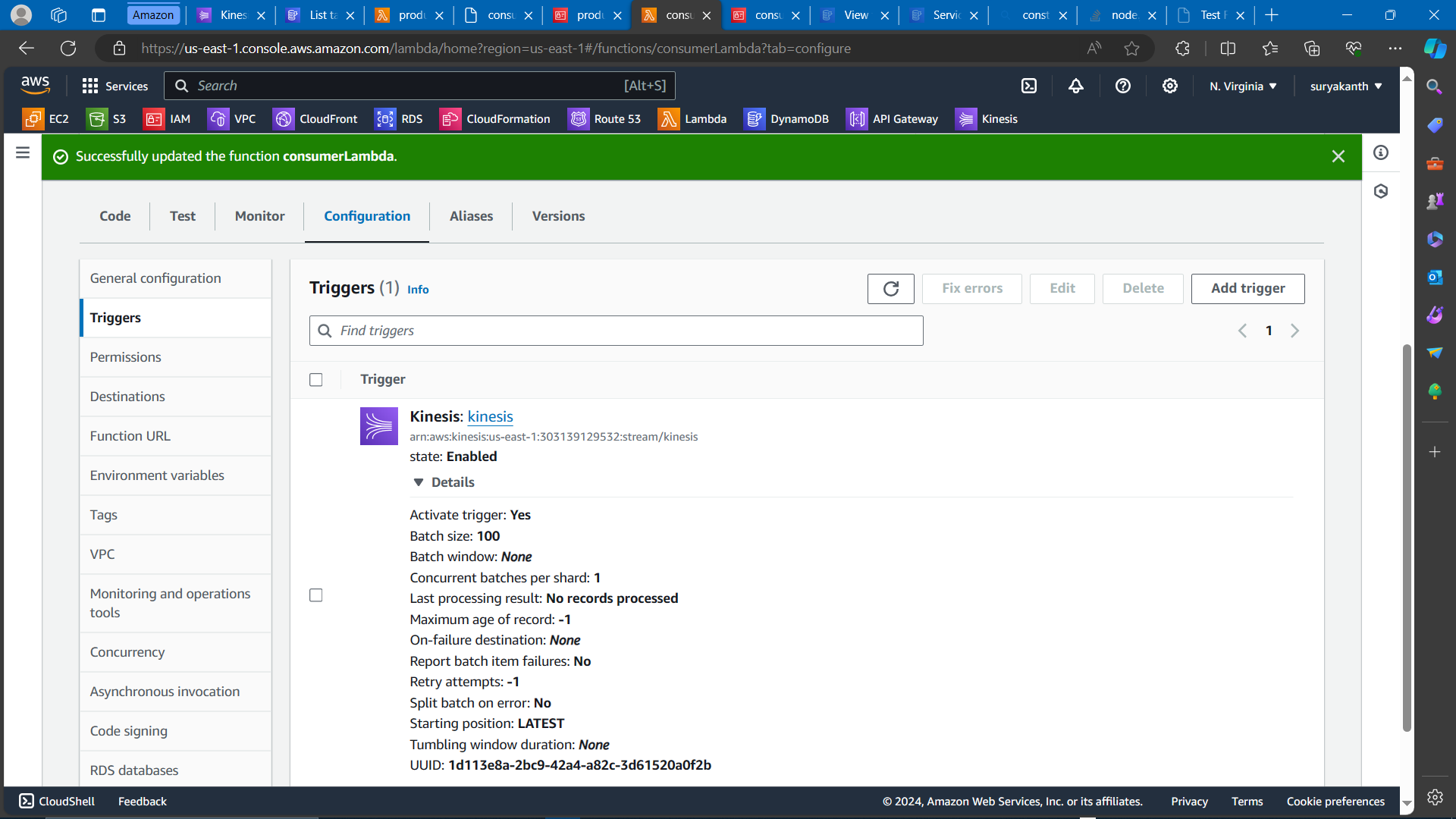


Fig 4 – Kinesis data stream which triggers consumer lambda function

1. **Configure Kinesis Trigger for Consumer Lambda:**
   * Navigate to the ConsumerLambda function in the Lambda service.
   * Click on "Add trigger."
   * Choose Kinesis as the trigger source.
   * Select the Kinesis Data Stream created earlier.
   * Configure batch size and starting position.
   * Click on "Add."

  
  
Fig 5 – Trigger configuration in consumer Lambda function using created kinesis

1. **Testing and Verification:**
   * Test the ProducerLambda function by invoking it manually or setting up a test event with sample data.
   * Monitor the Kinesis Data Stream to ensure that data is being continuously sent.
   * Verify that the ConsumerLambda function is triggered by the incoming data in the Kinesis stream.
   * Check the DynamoDB table to confirm that the data is being stored correctly.
   * Use CloudWatch logs and metrics to troubleshoot any issues and monitor the performance of the Lambda functions and Kinesis Data Stream.
   * Perform load testing to ensure that the system can handle the expected volume of data.
   * Implement error handling and retry mechanisms to handle any failures gracefully.
   * Conduct end-to-end testing to validate the entire workflow, including data ingestion, processing, and storage.

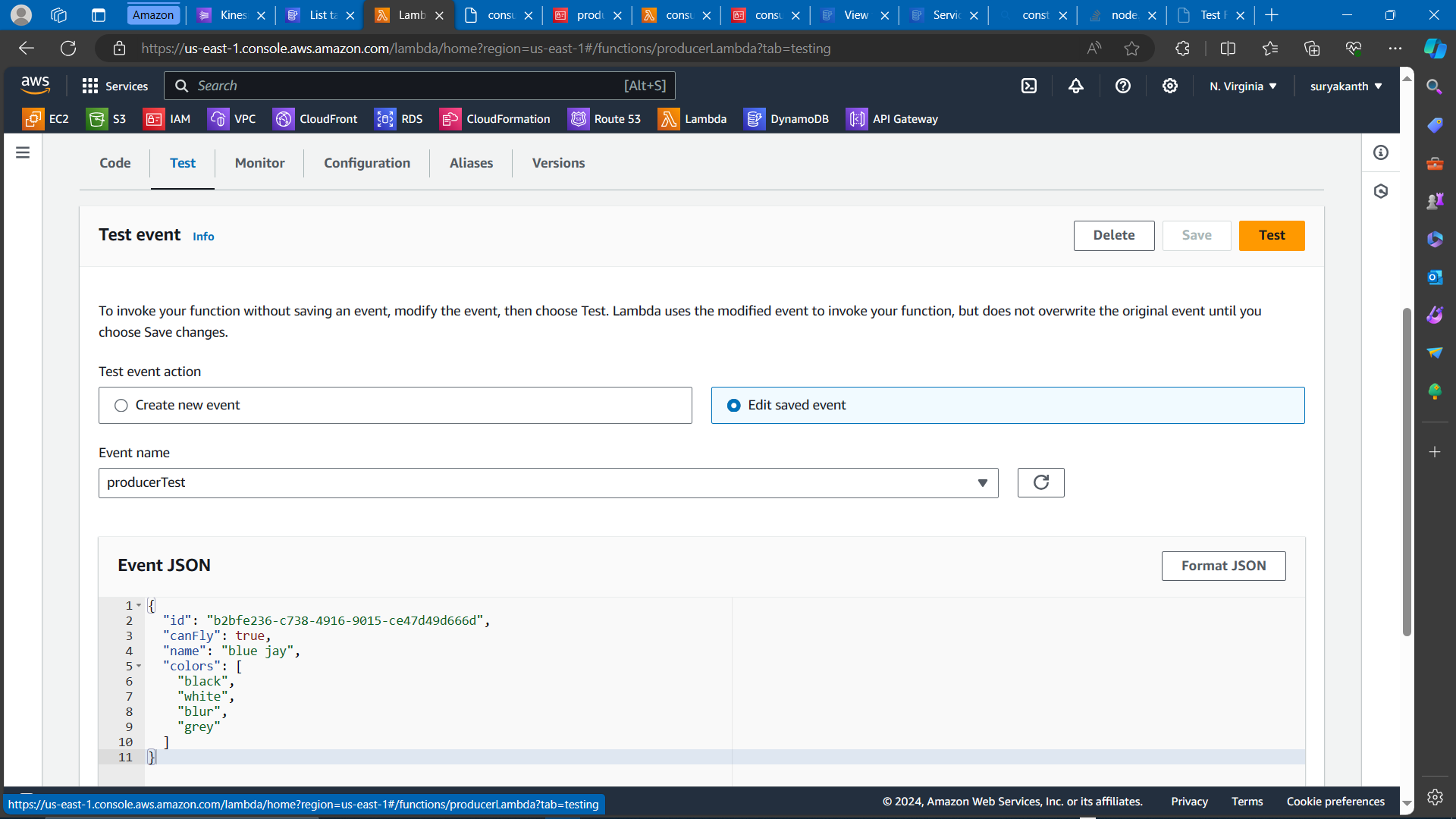
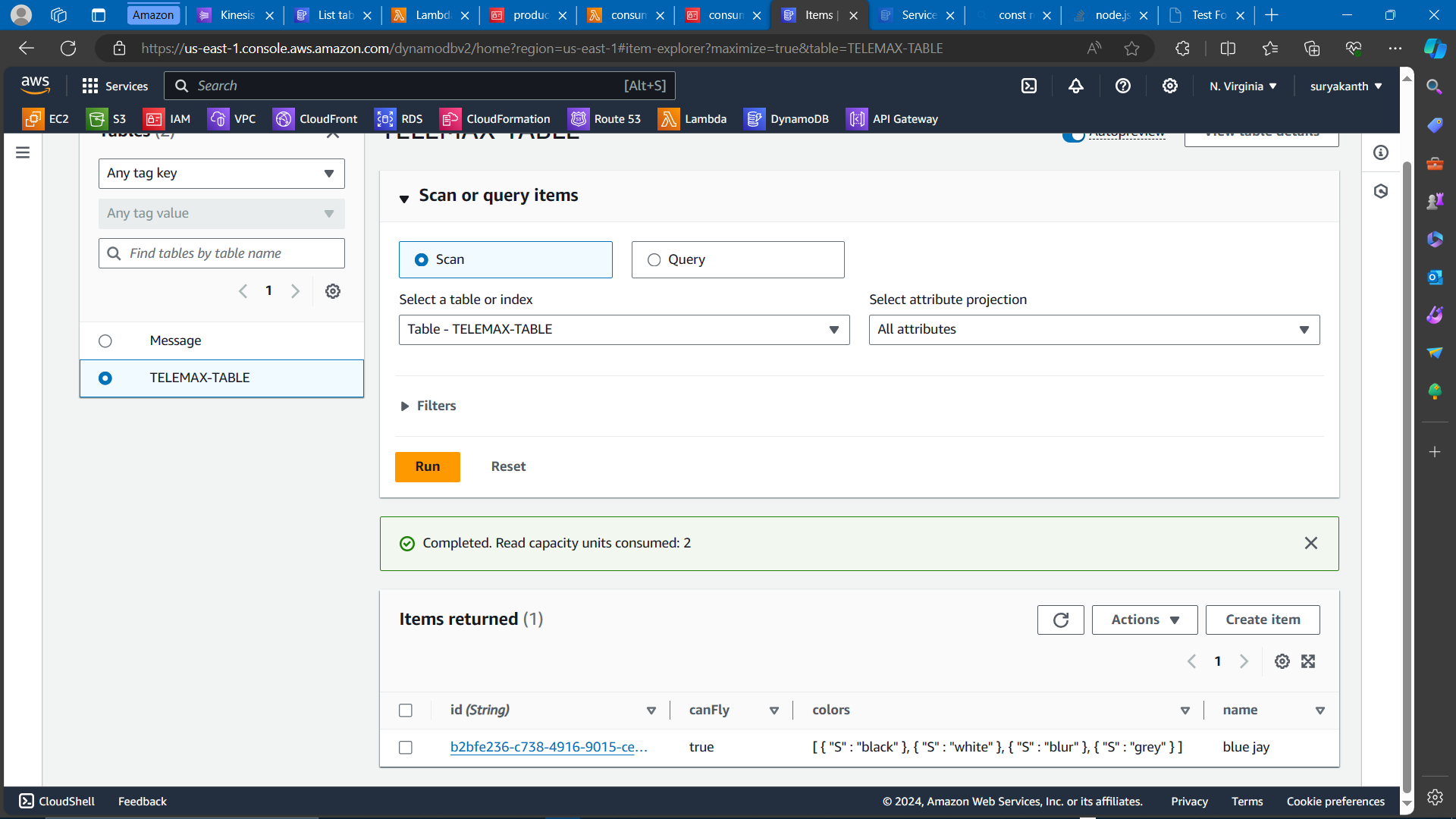


Fig 6 – Testing producer lambda function with sample data  
  
  
  
Fig 7 – Sample data saved in dynamoDB table

**Project Completion Note:**

"Real-Time Data Processing with Serverless Architecture on AWS" project has been completed successfully. Leveraging AWS services, we've implemented a scalable, flexible, and cost-efficient serverless architecture for real-time data processing. The solution includes an empty DynamoDB table and two Lambda functions: a producer lambda for data ingestion and a consumer lambda for processing and storing data. This architecture empowers rapid decision-making and real-time analytics across diverse industries.