CLOUD & SERVERLESS COMPUTING

FINAL PROJECT DOCUMENTATION

NAME : KADAVALA VANI VENKATA DURGA

REG NO :2000031863

TITLE: Build a Serverless Real-time Data Processing App

LINKEDIN LINK :

https://www.linkedin.com/pulse/build-serverless-real-time-data-processing-app-vani-kadavala/?trackingId=2Ltj1nzec1R6okt9ZnrJTg%3D%3D

GITHUB LINK : https://github.com/vanikadavala/PE3PROJECT

VIDEO LINK :

https://drive.google.com/file/d/1tyG9FDm9Bpuzpd76MFHMqEKAD2XEf5x9/view?usp=sharing

INTRODUCTION :

In this project, we'll learn how to build a serverless app to process real-time data streams. we'll create software using Amazon to instantly process and display this data. we'll process real-time streams using AWS Lambda, store records in a NoSQL database using Amazon DynamoDB, aggregate data using Amazon Kinesis Data Analytics, archive raw data to Amazon S3 using Amazon Kinesis Data Firehose, and perform ad-hoc queries on the raw data using Amazon Athena.

SERVICES USED :

Amazon Athena

Amazon Kinesis Data Firehose

Amazon DynamoDB

Amazon S3

Amazon IAM

AWS Lambda

Amazon Cognito

Amazon Kinesis Data Analytics

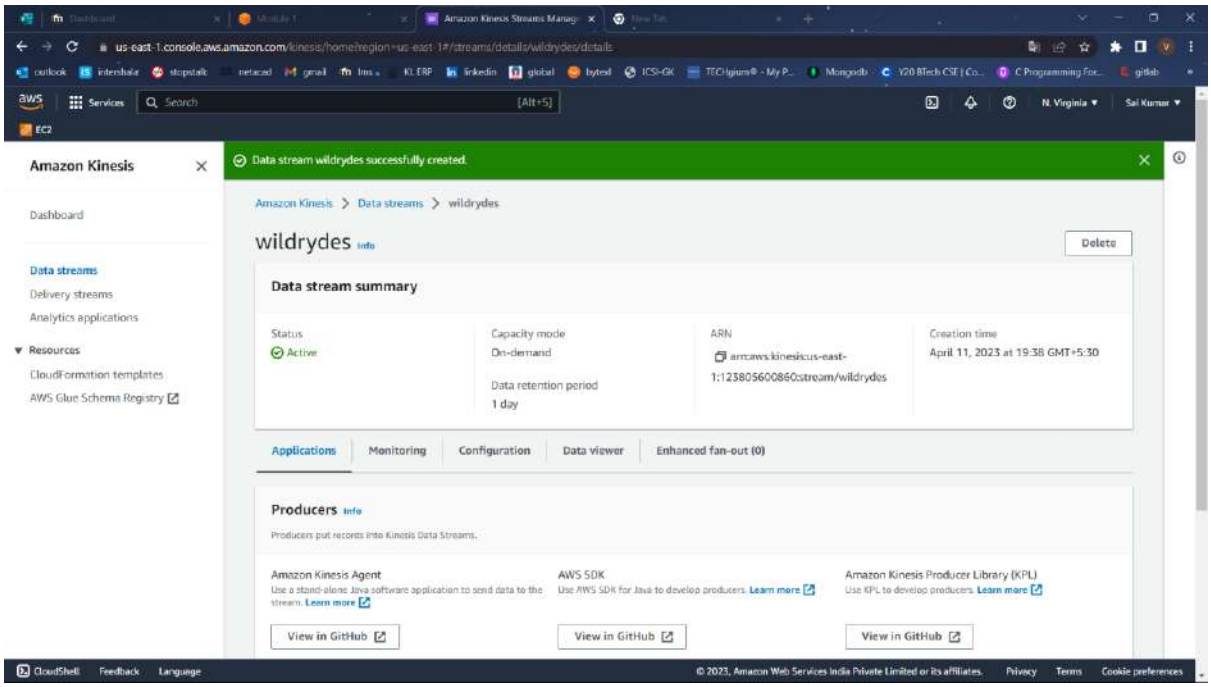
Amazon Kinesis Data Streams

IMPLEMENTATION :

STEP 1 : Build a data stream

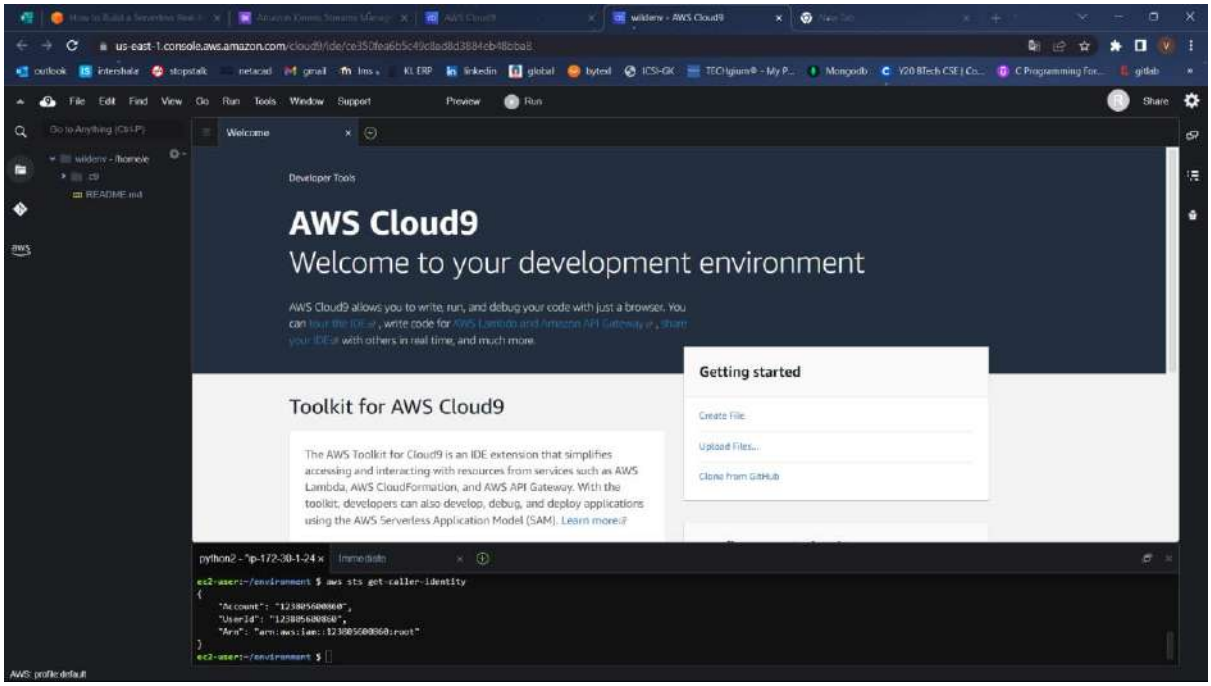
→ Create an Amazon Kinesis stream

create a new stream named *wildrydes* with *1* shard



your Kinesis stream will be ACTIVE and ready to store real-time streaming data

→Produce messages into the stream



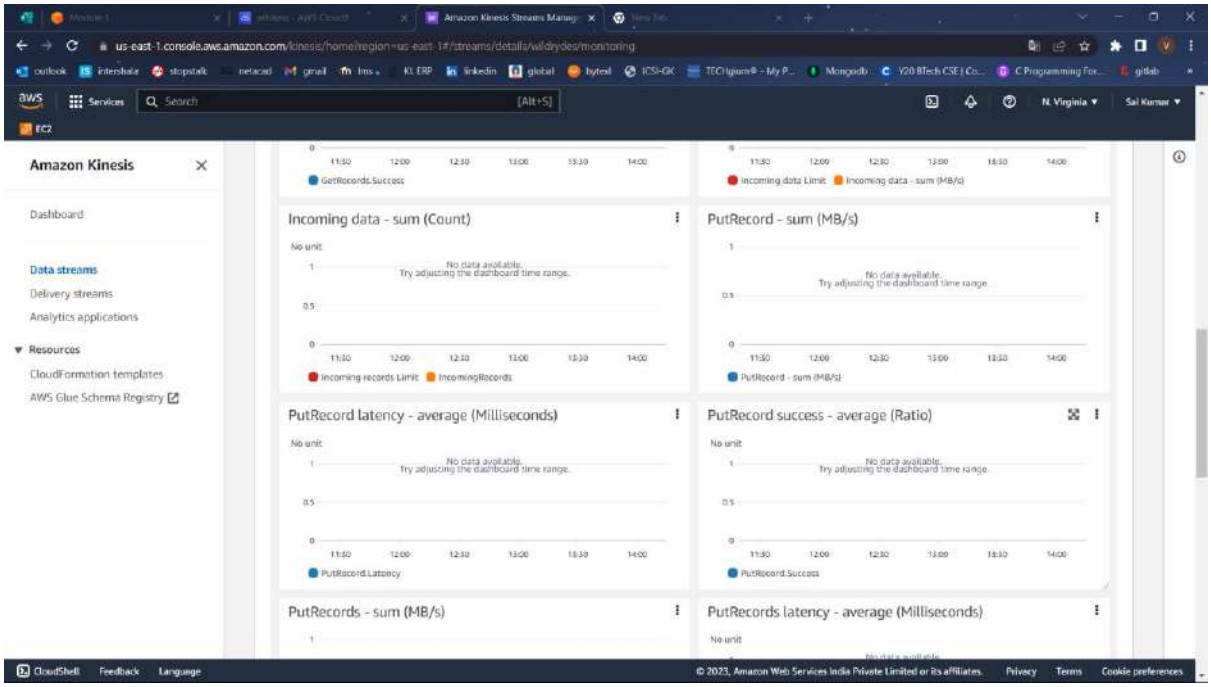
Activate cloud9 environment

In the terminal, run the producer to start emiting sensor data to the stream

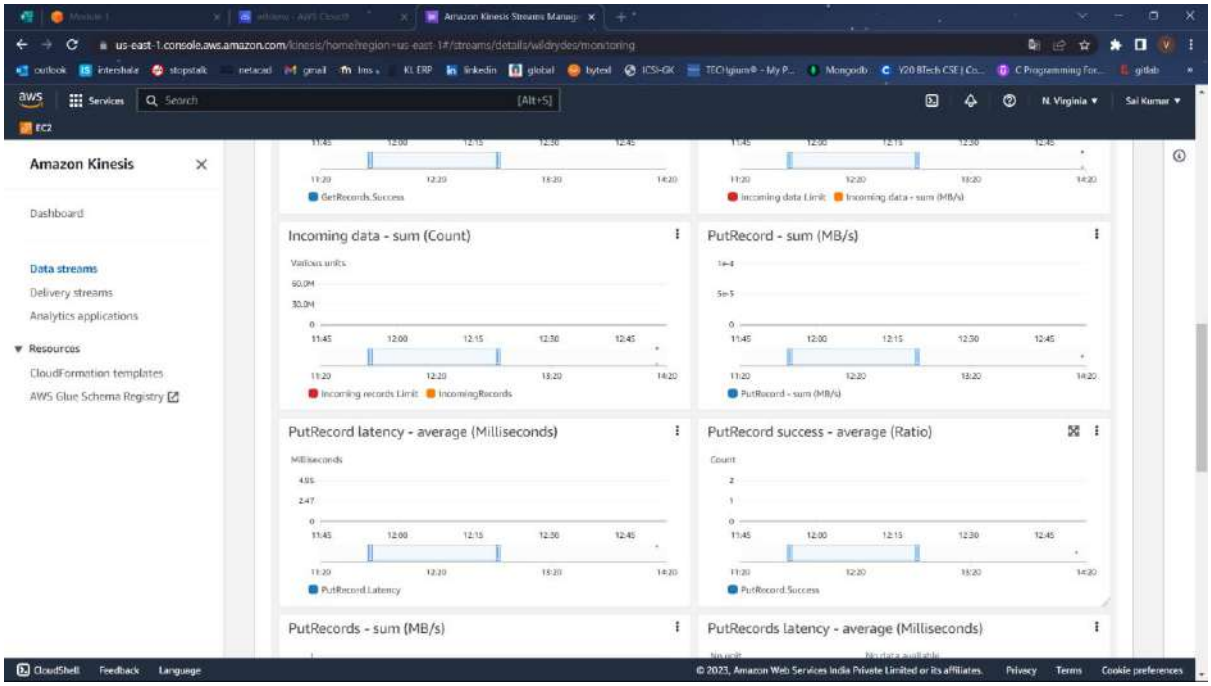
In the Amazon Kinesis Streams console, click on wildrydes and click on the Monitoring tab.

After several minutes, you will see the Put Record (Bytes) — Sum graph begin to record several thousand bytes put per minute

BEFORE-

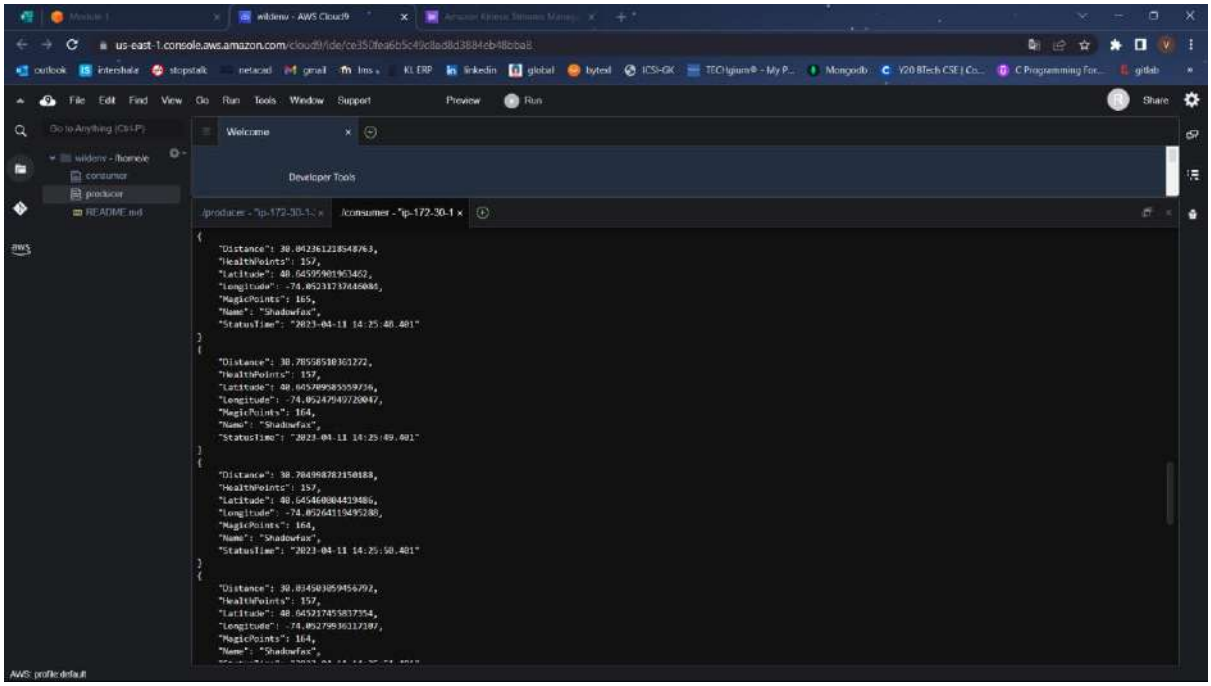


AFTER-

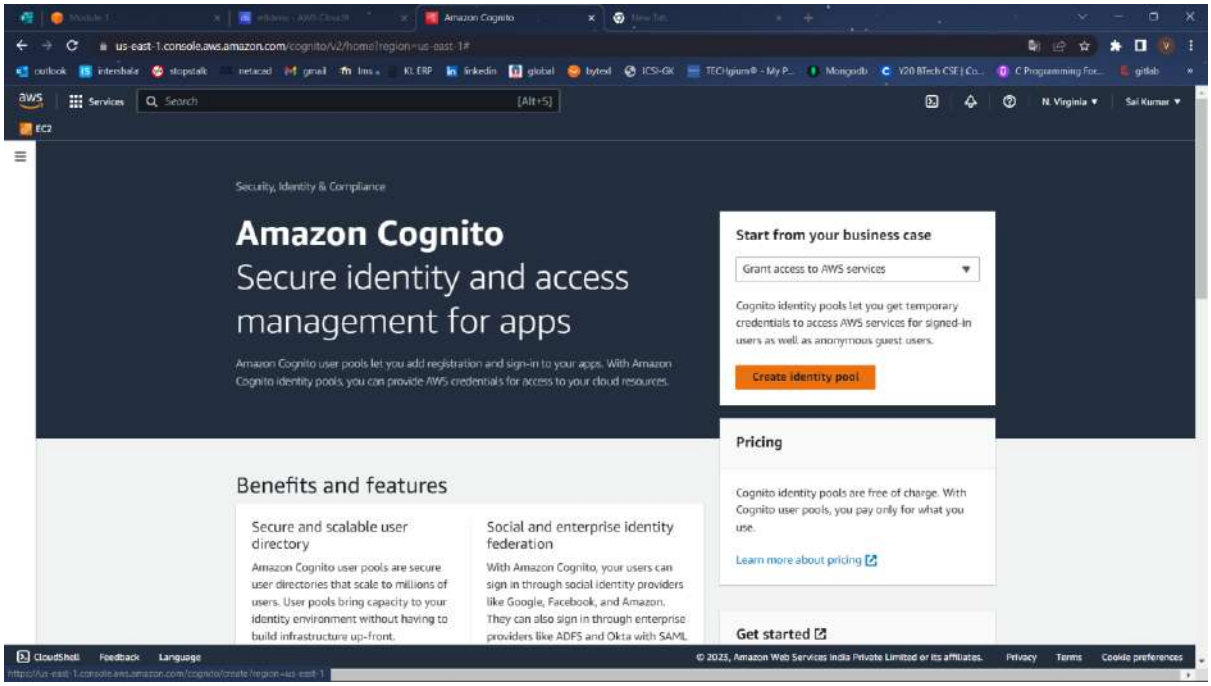


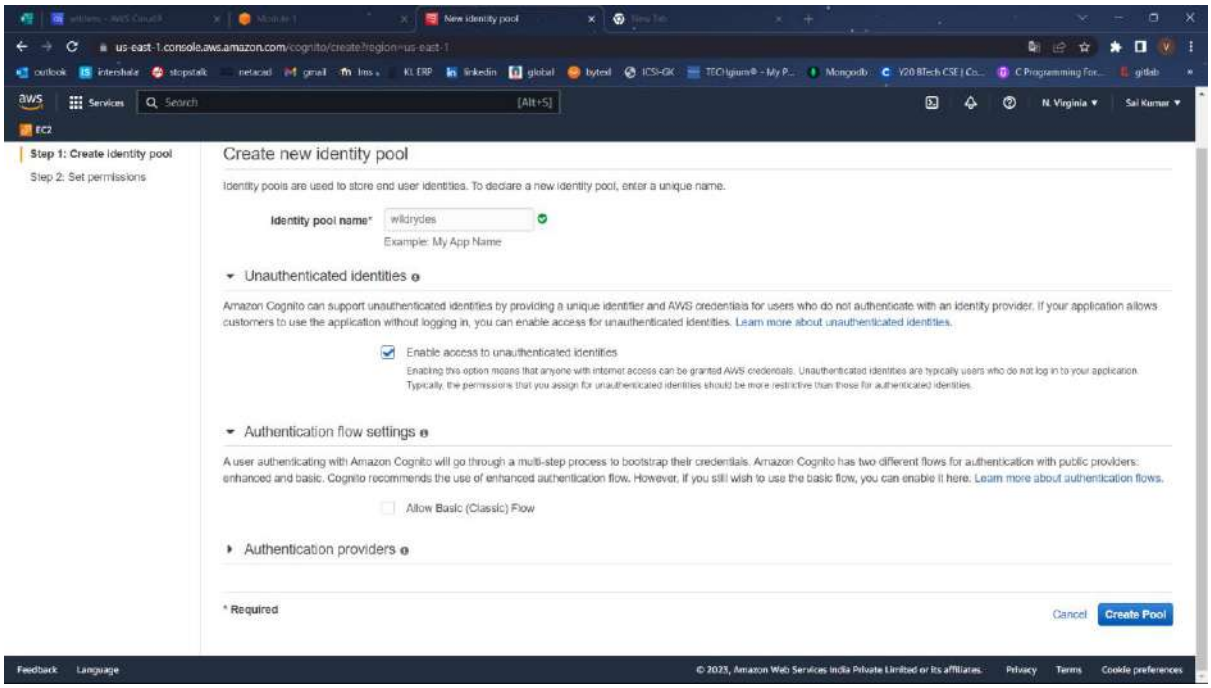
Below is the message stored from the above graph

consumer will print the messages being sent by the producer

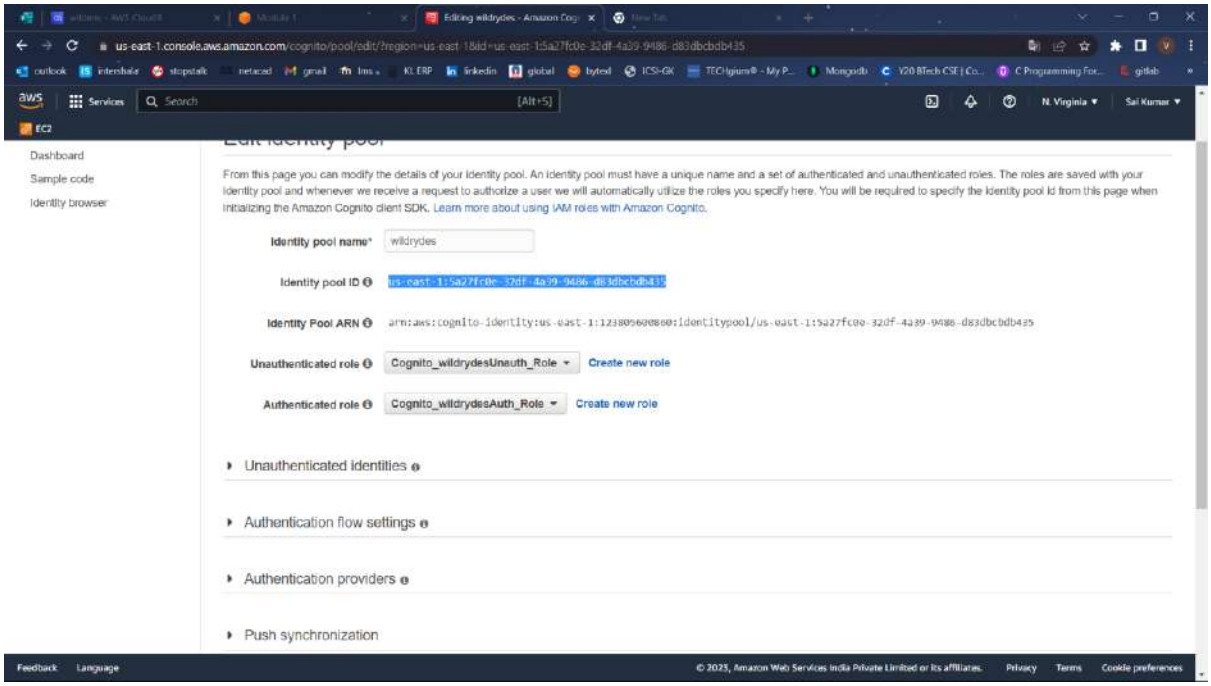


→ Create an identity pool for the unicorn dashboard

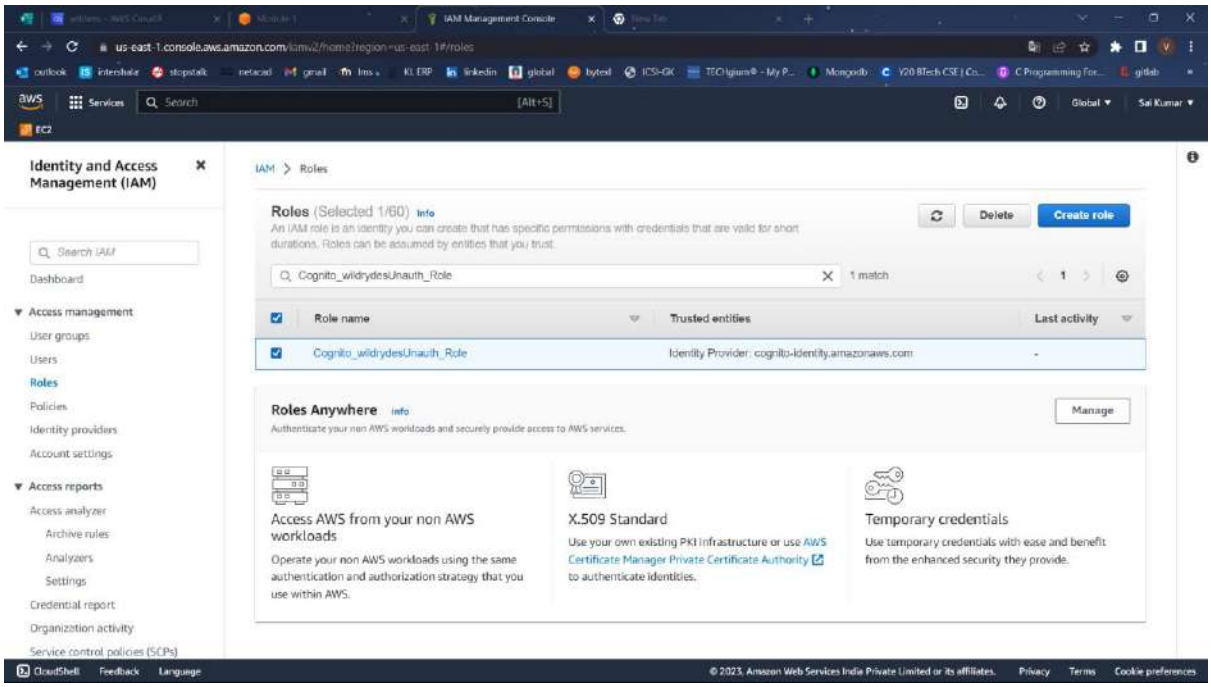




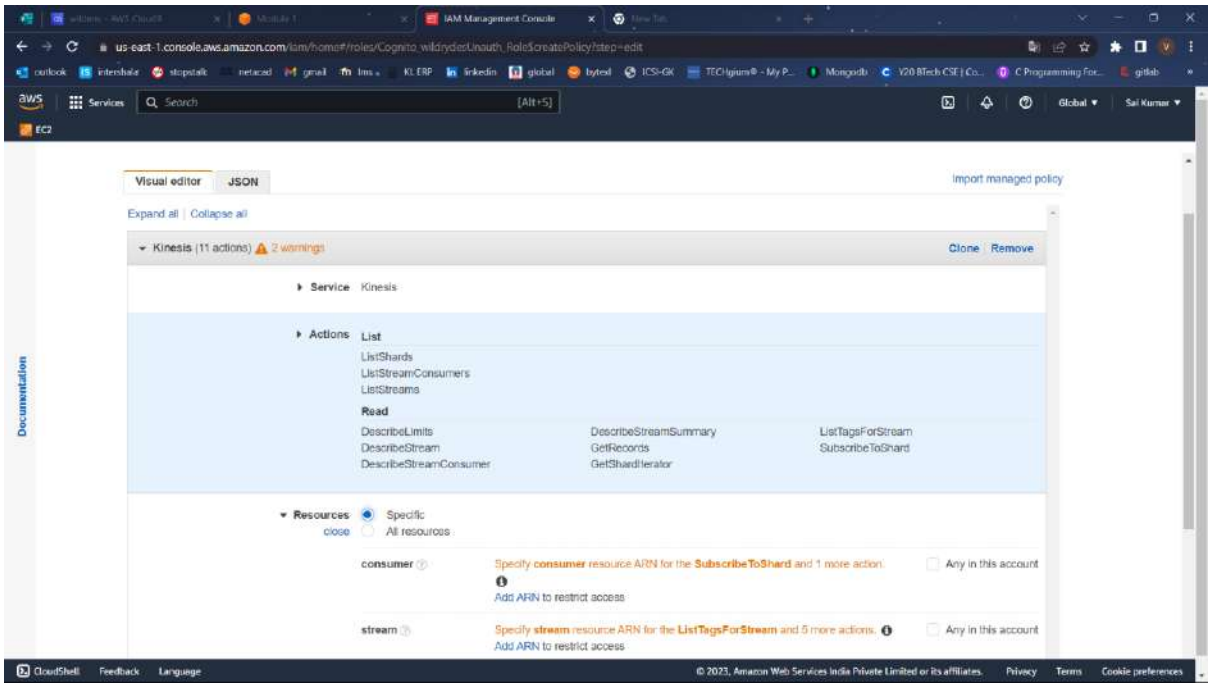
Note identity pool id- `us-east-1:5a27fc0e-32df-4a39-9486-d83dbcbdb435`

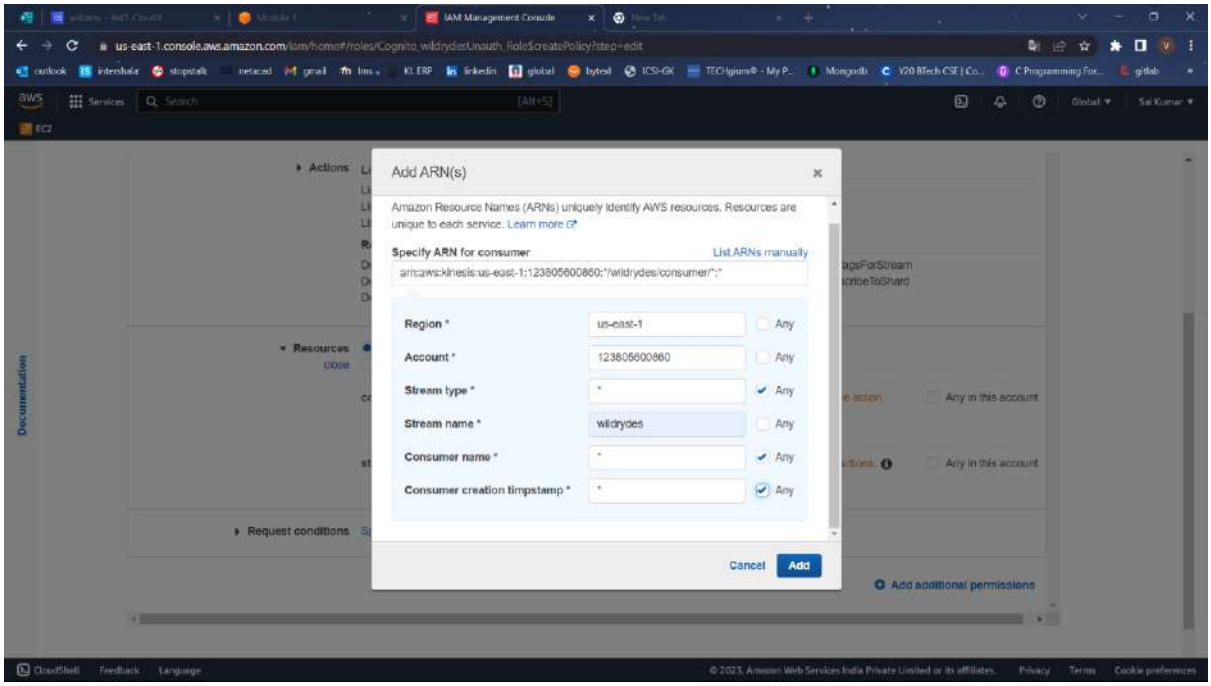


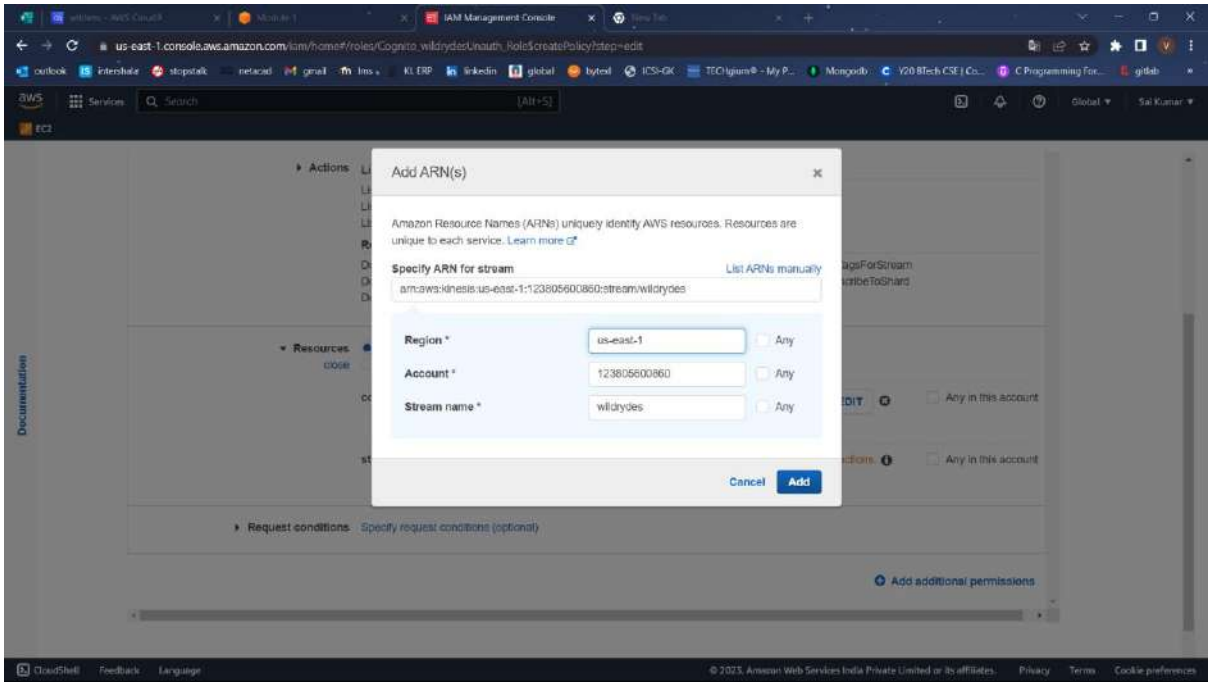
→ Grant the unauthenticated role access to the stream

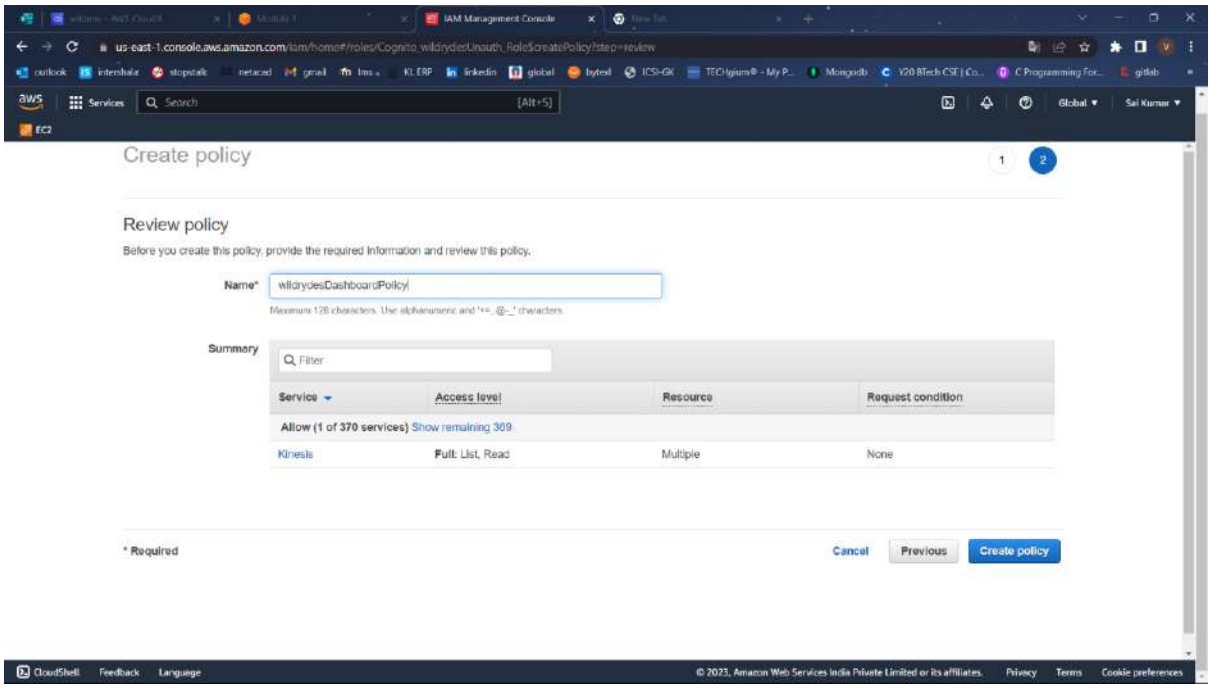


Create inline policy





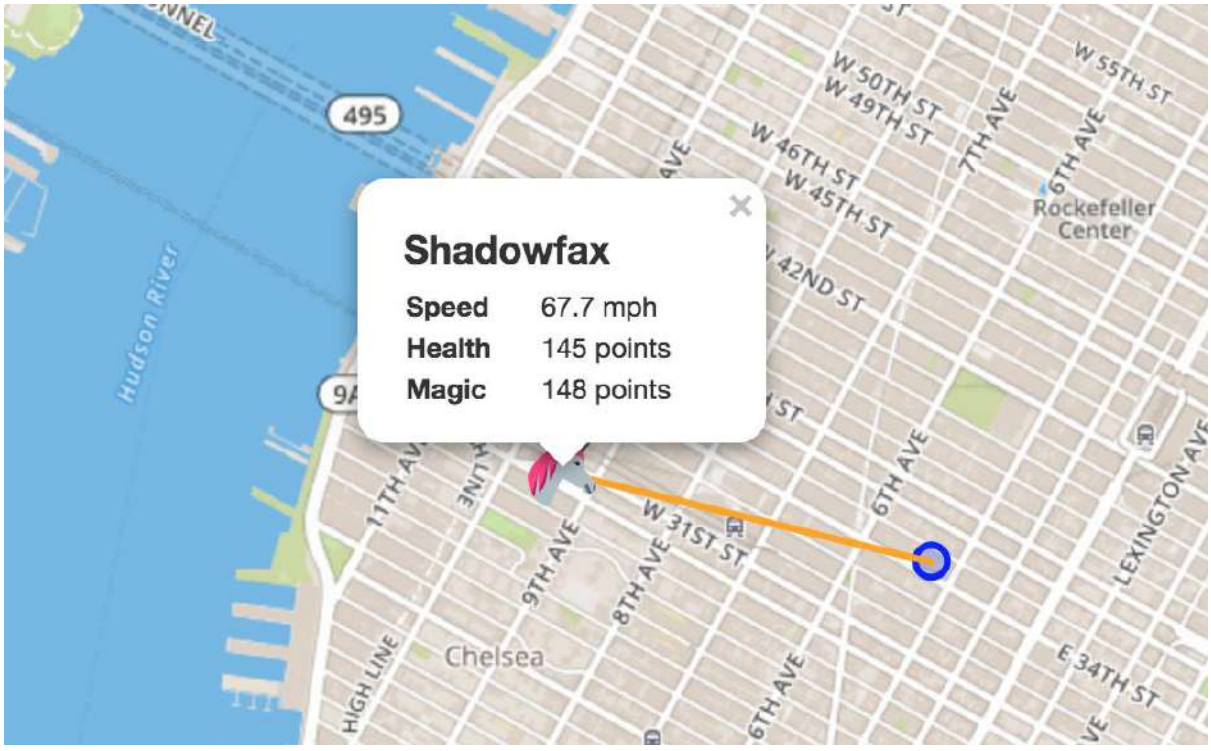




Policy created.

→ View unicorn status on the dashboard

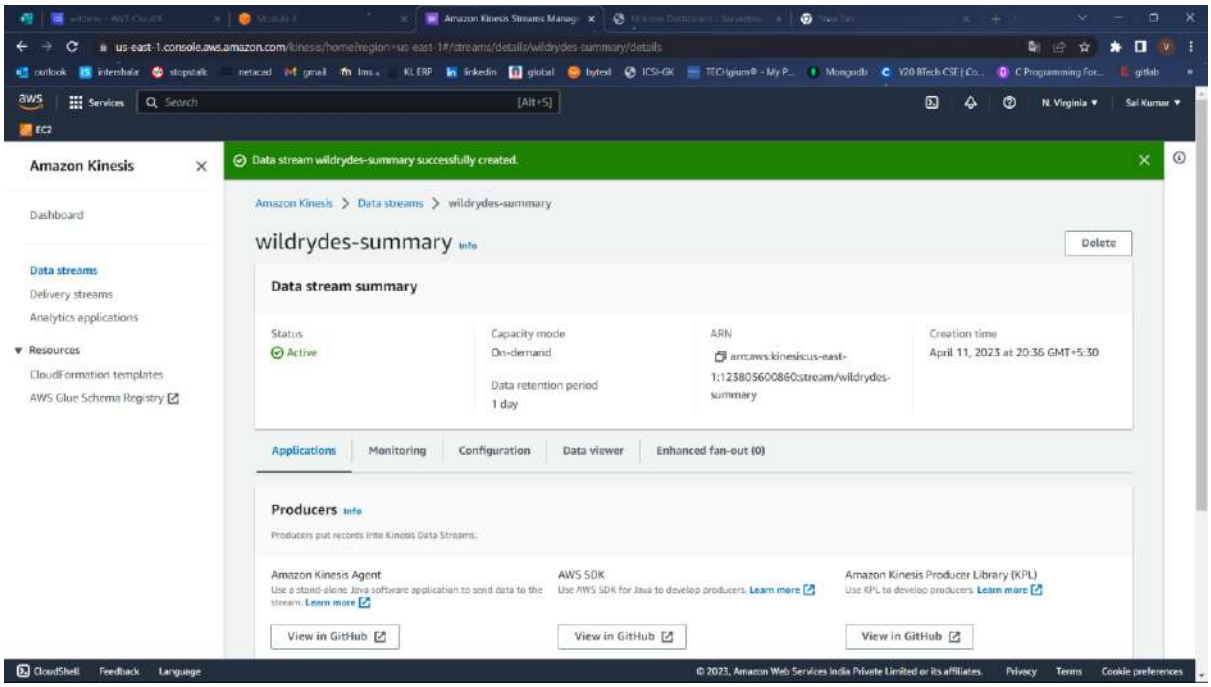
Validate that you can see the unicorn on the map



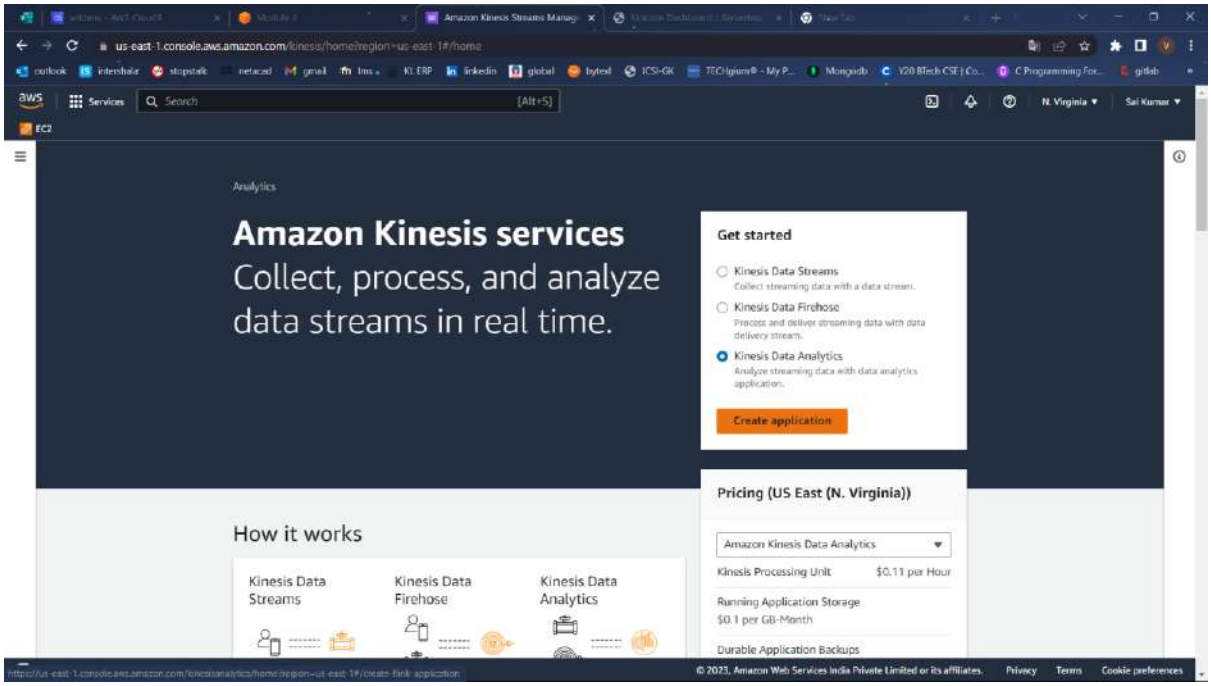
STEP 2: Aggregate data

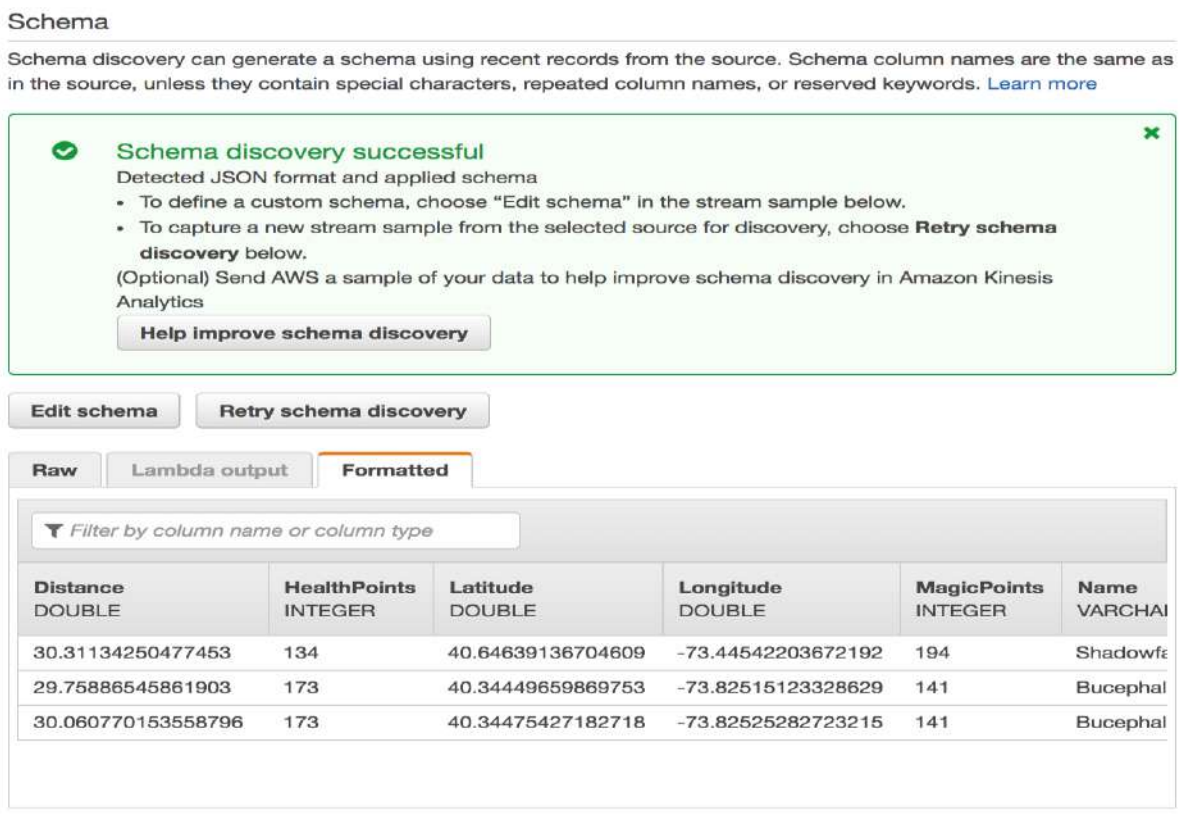
→ Create an Amazon Kinesis stream

Create another stream



→ Create an Amazon Kinesis Data Analytics application



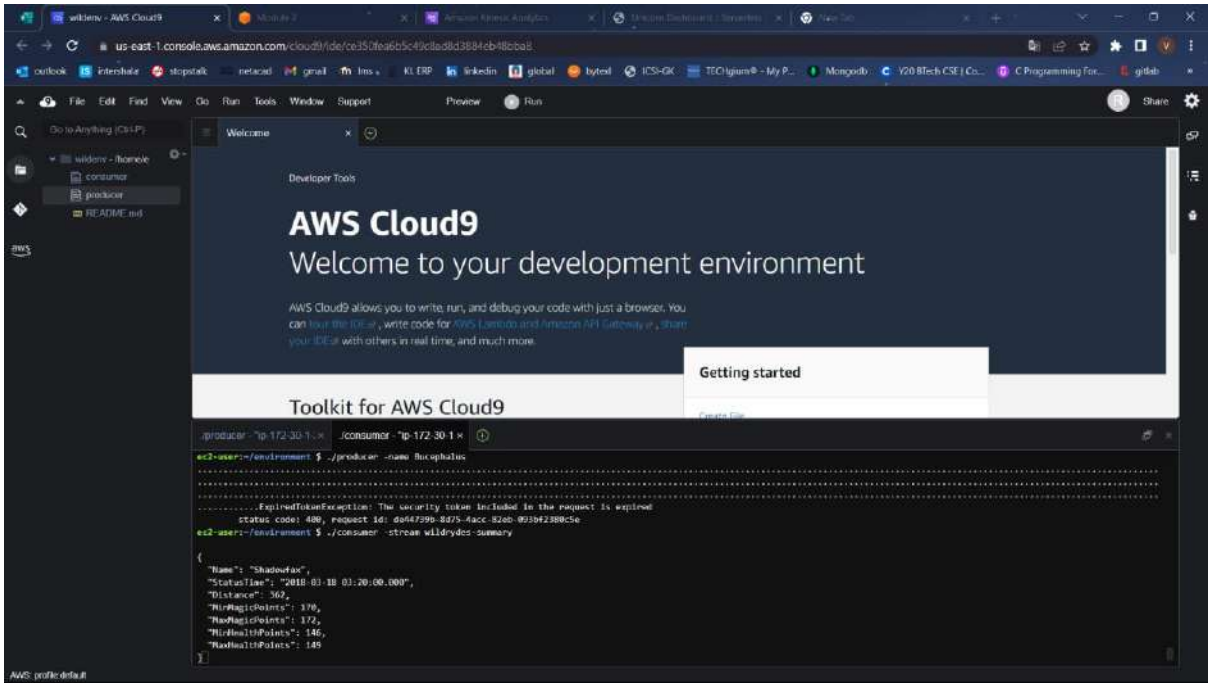


Schema created in the data analytics stream

→ Read messages from the stream

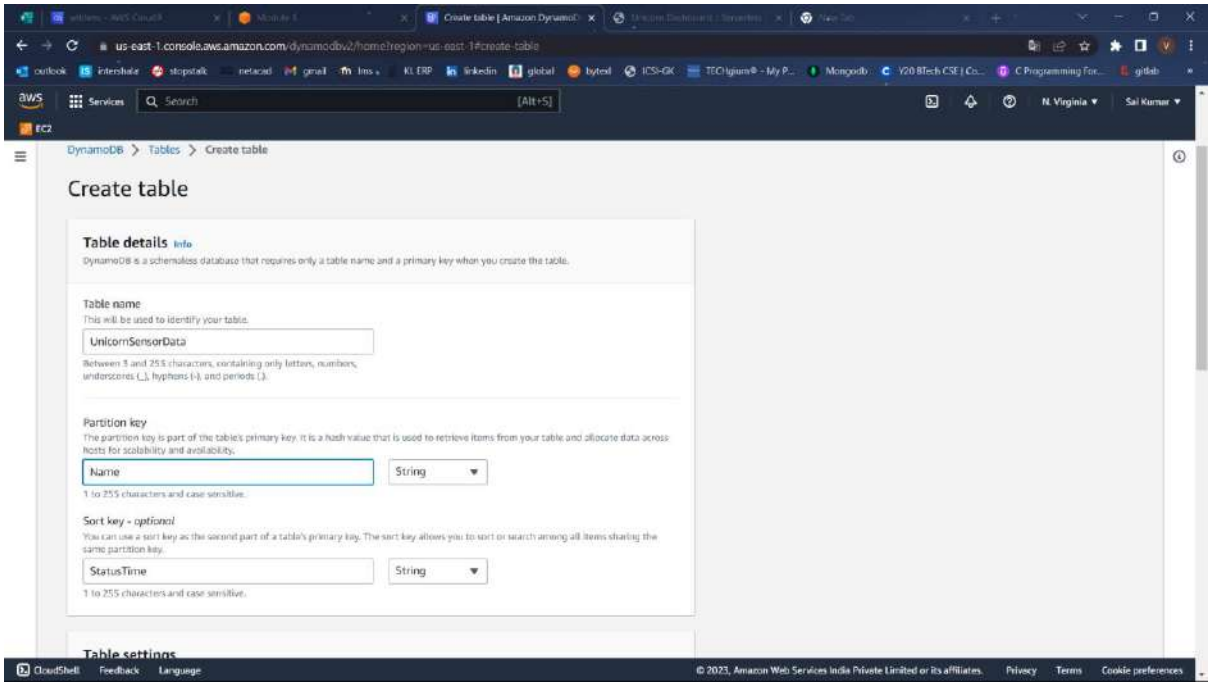
Run command - ./consumer -stream wildrydes-summary

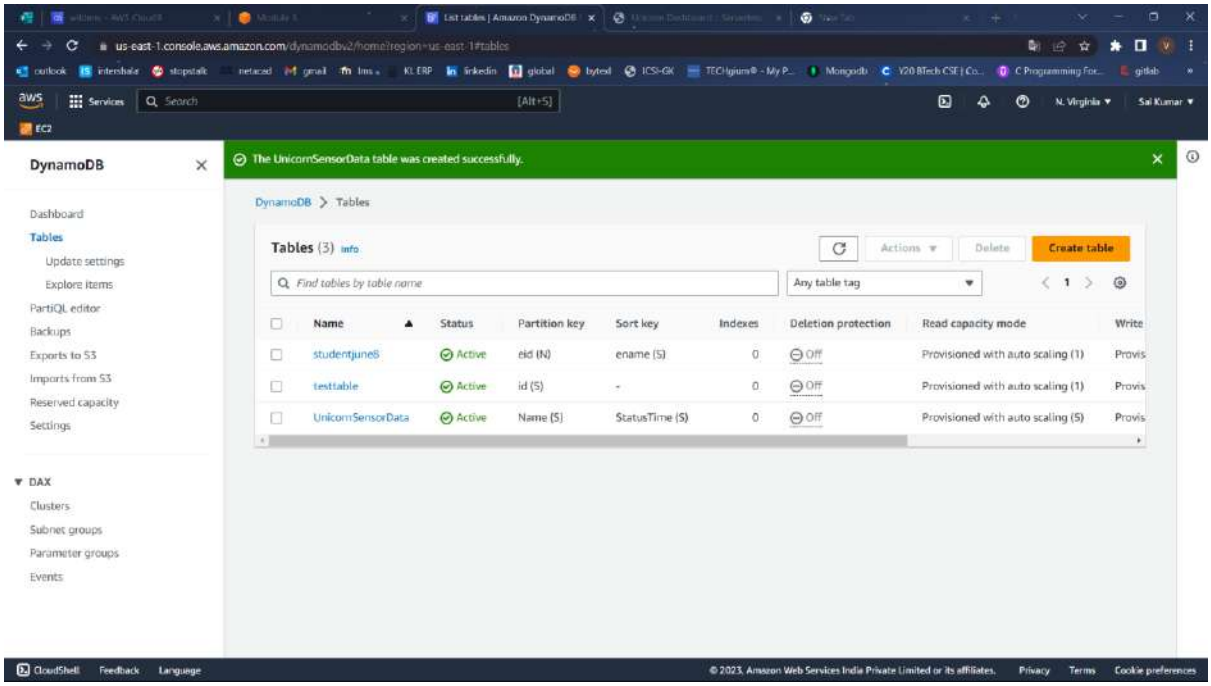
→following output should be reflected , it seems that data is stored successfully



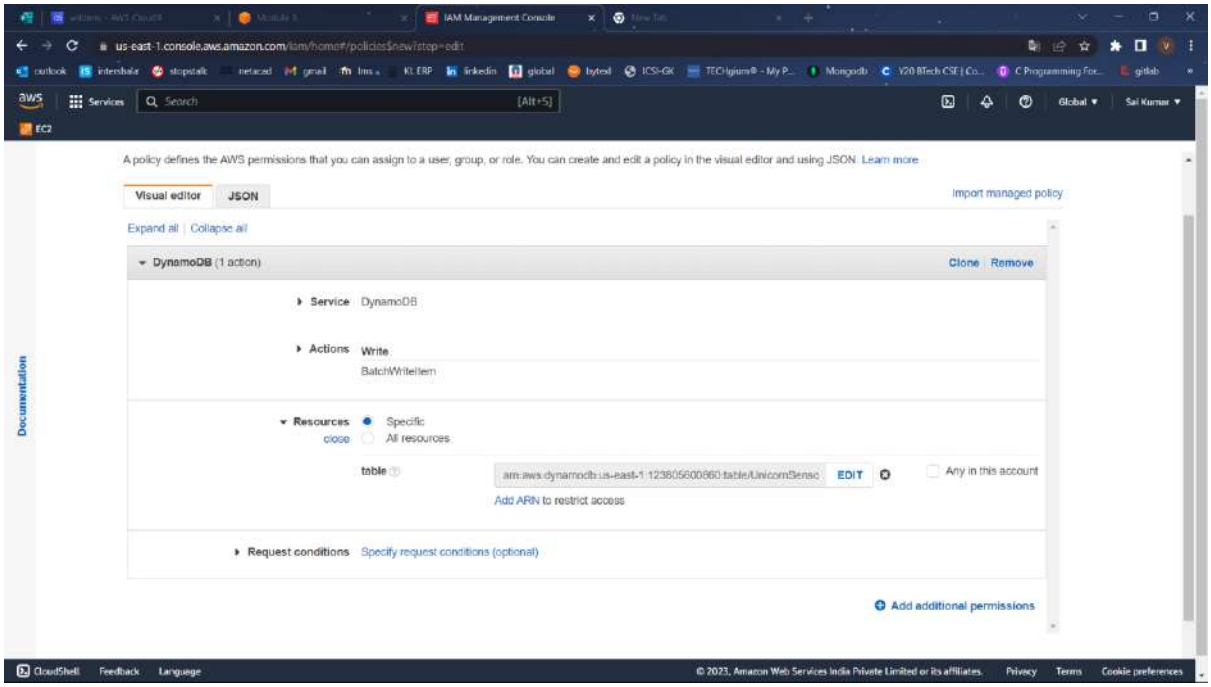
STEP 3 : Process streaming data

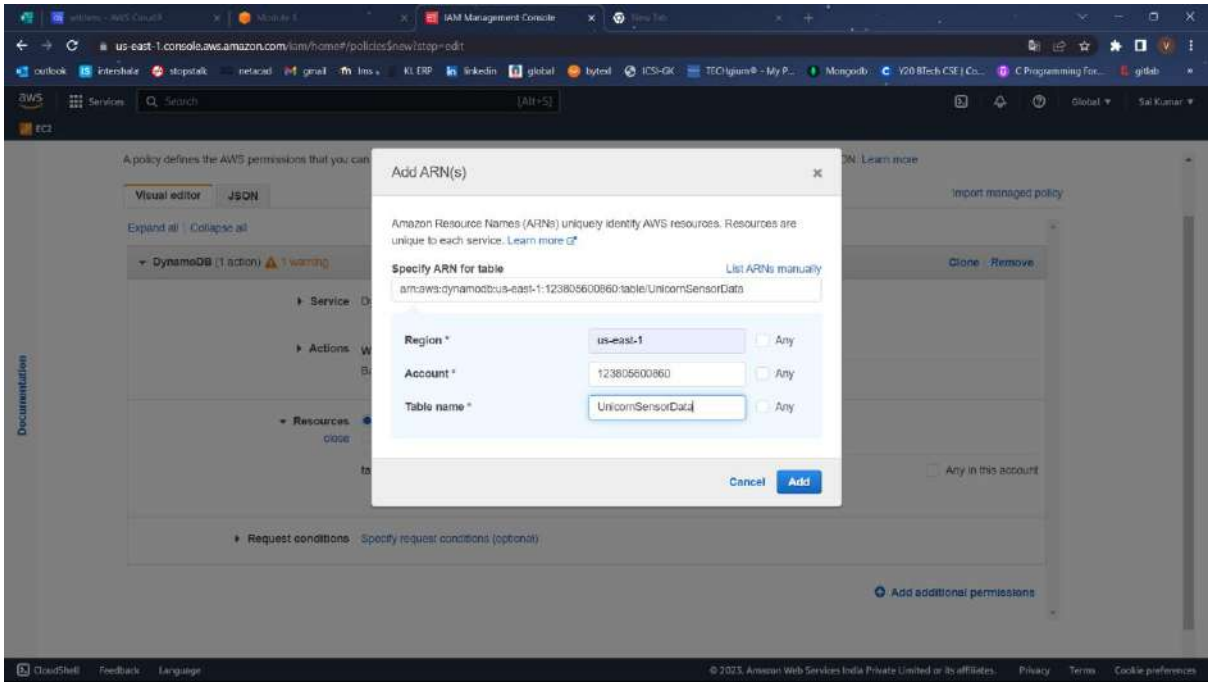
→ Create an Amazon DynamoDB table

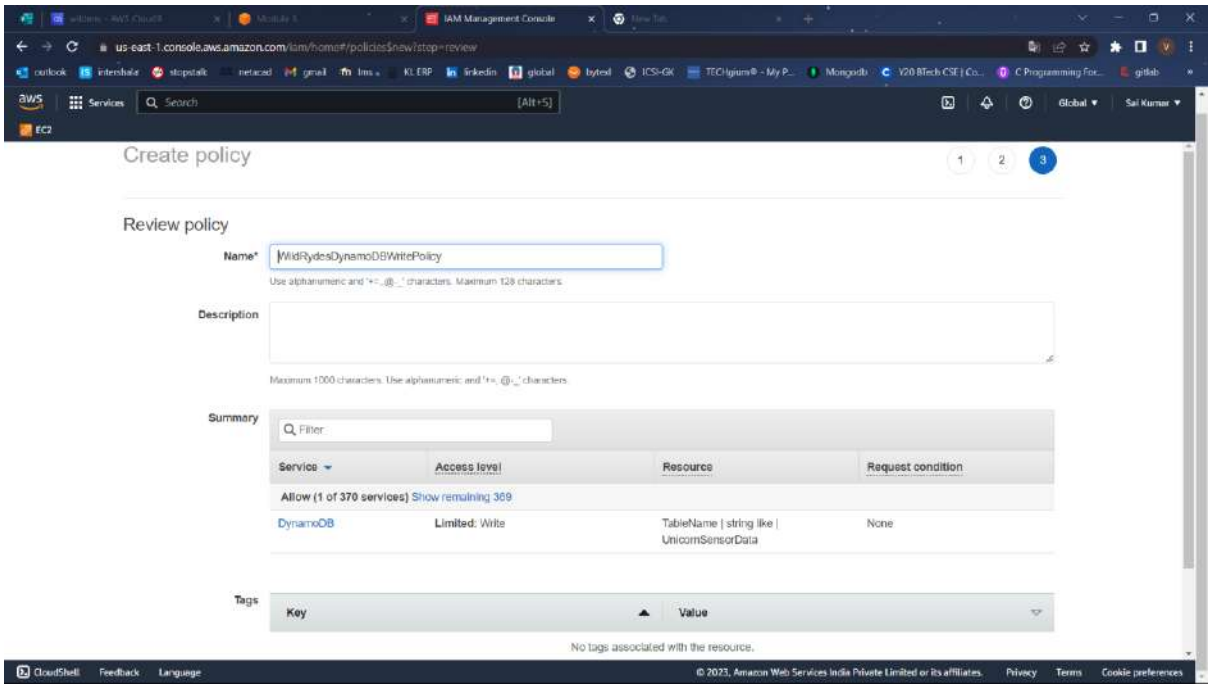


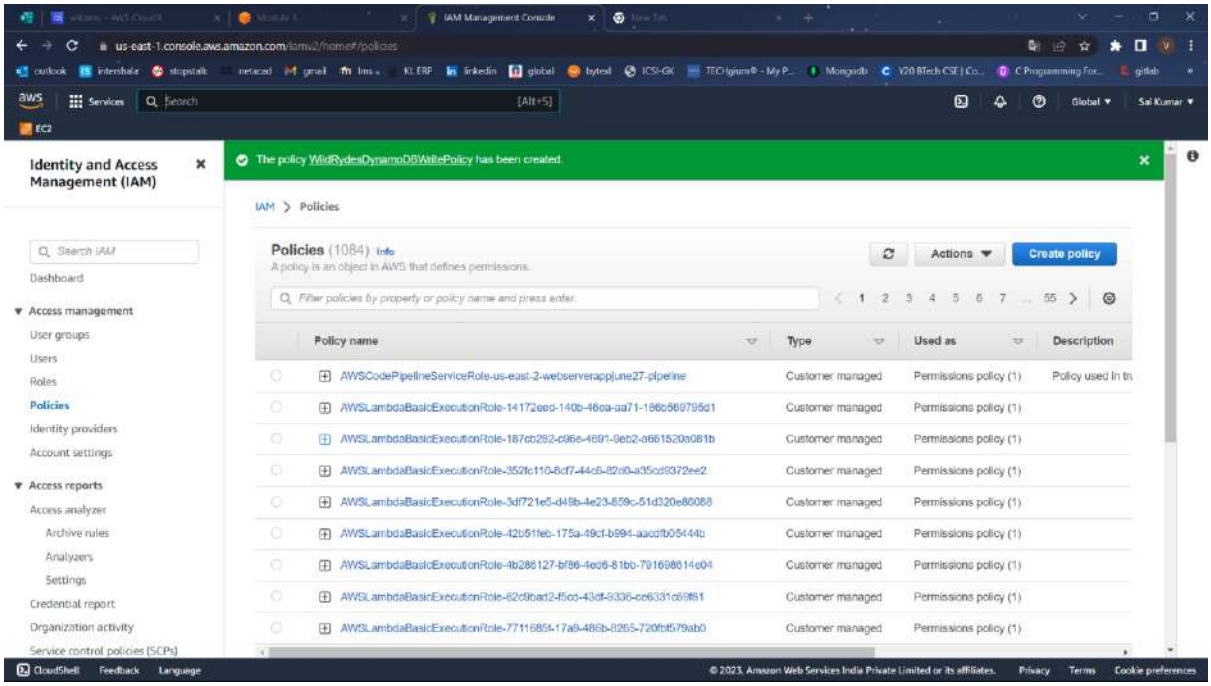


→ Create an IAM role for your Lambda function

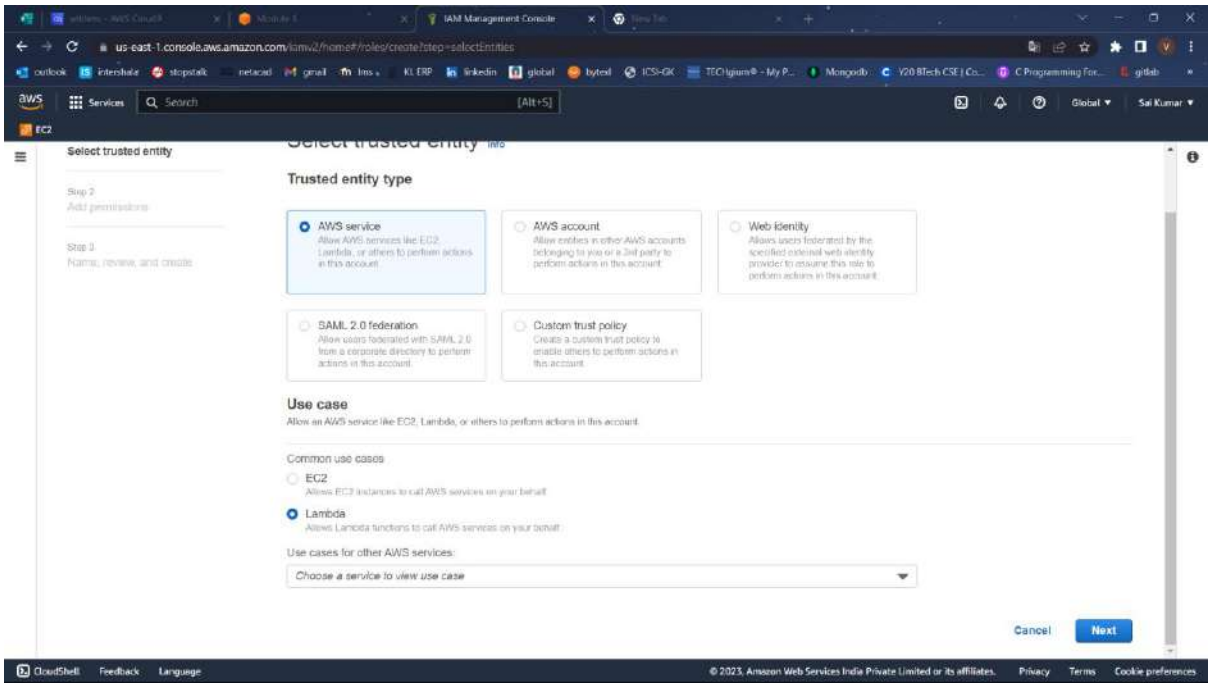


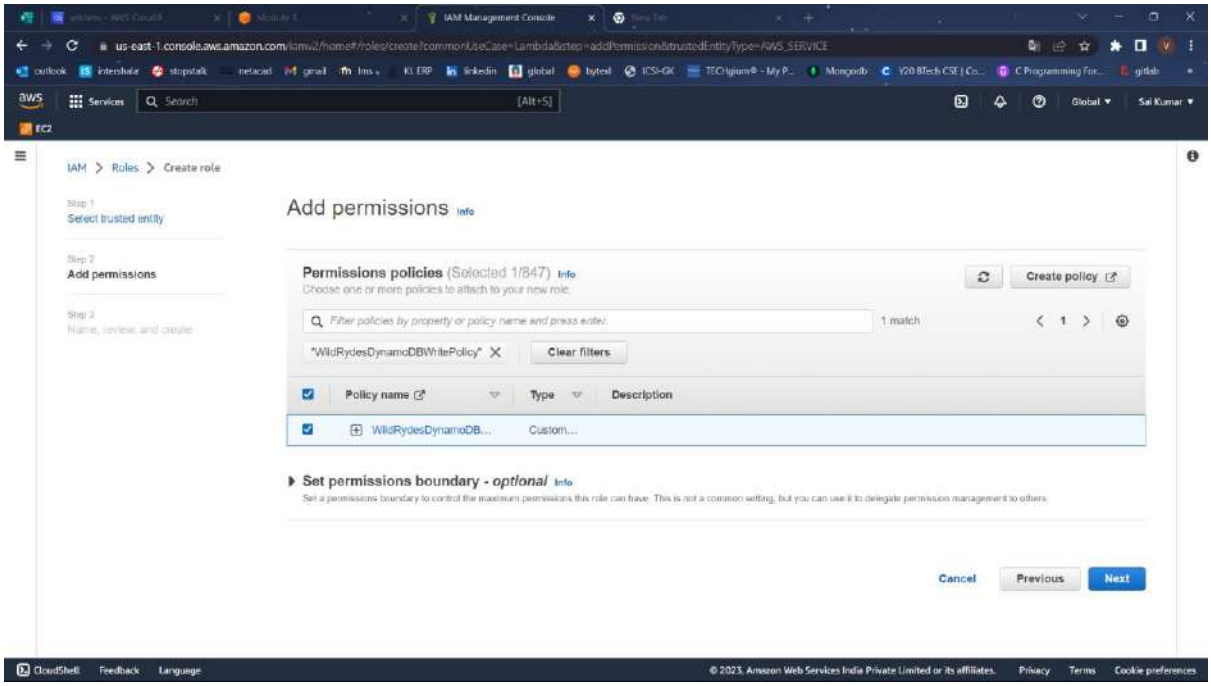

Add ARN(s)
Amazon Resource Names (ARNs) uniquely identify AWS resources. Resources are unique to each service. Learn more
Specify ARN for table
List ARNs manually
arn:aws:dynamodb:us-east-1:123805600860:table/UnicornSensorData
Region *
us-east-1
Any
Account *
123805600860
Any
Table name *
UnicornSensorData
Any
Cancel
Add
Visual editor
JSON
Import managed policy
Expand all | Collapse all
DynamoDB (1 action) 1 warning
Clone
Remove
Service
Actions
Resources
close
Any in this account
Request conditions
Specify request conditions (optional)
Add additional permissions
Documentation
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

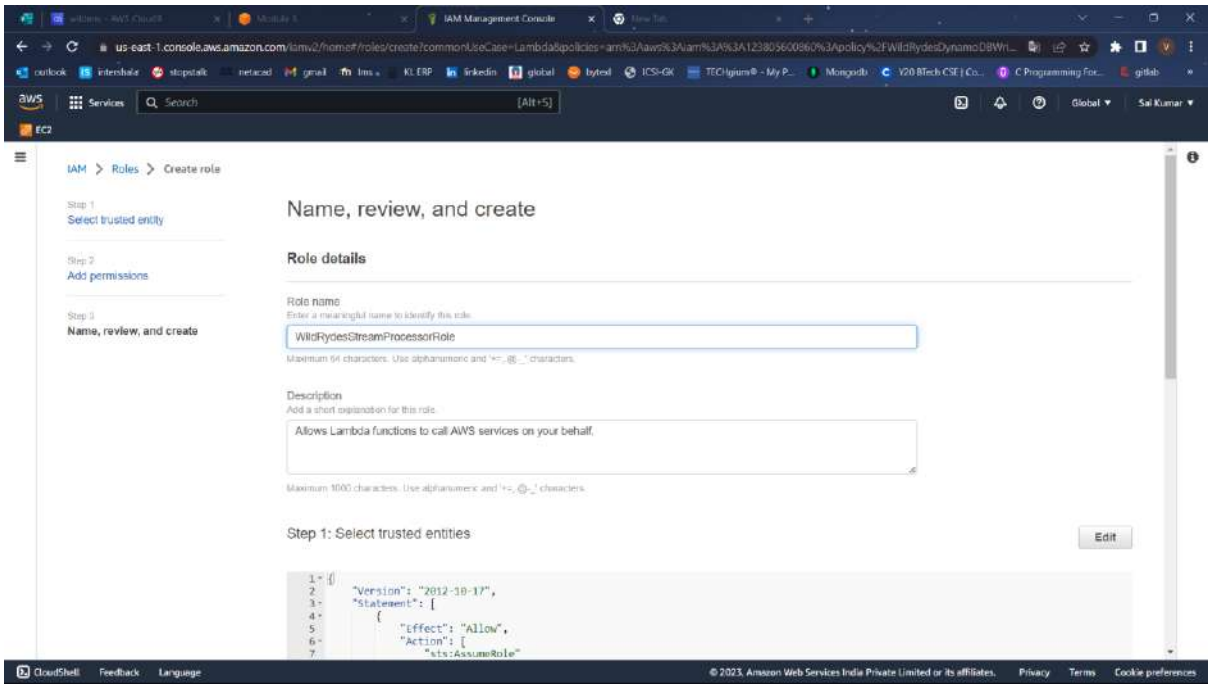

Create policy
1
2
3
Review policy
Name*
WildRydesDynamoDBWritePolicy
Use alphanumeric and '+=,.@_-' characters. Maximum 128 characters.
Description
Maximum 1000 characters. Use alphanumeric and '+=,.@_-' characters.
Summary
Filter
Service
Access level
Resource
Request condition
Allow (1 of 370 services) Show remaining 369
DynamoDB
Limited: Write
TableName | string like | UnicornSensorData
None
Tags
Key
Value
No tags associated with the resource.
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

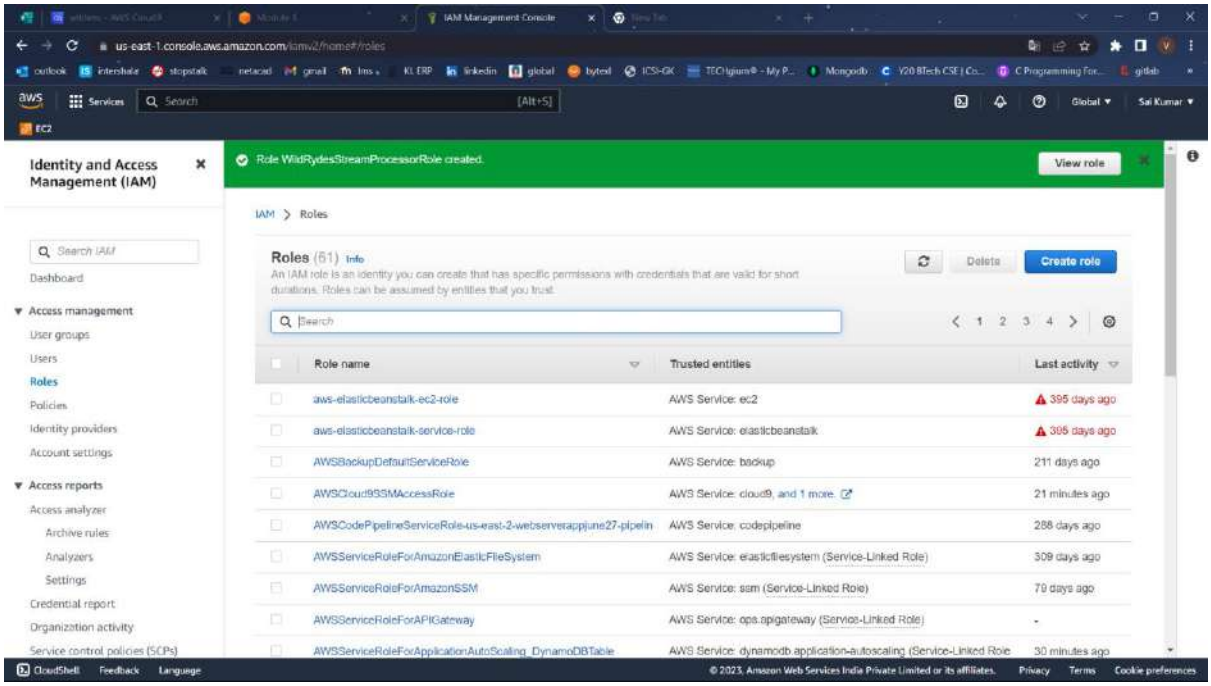


Policy created successfully



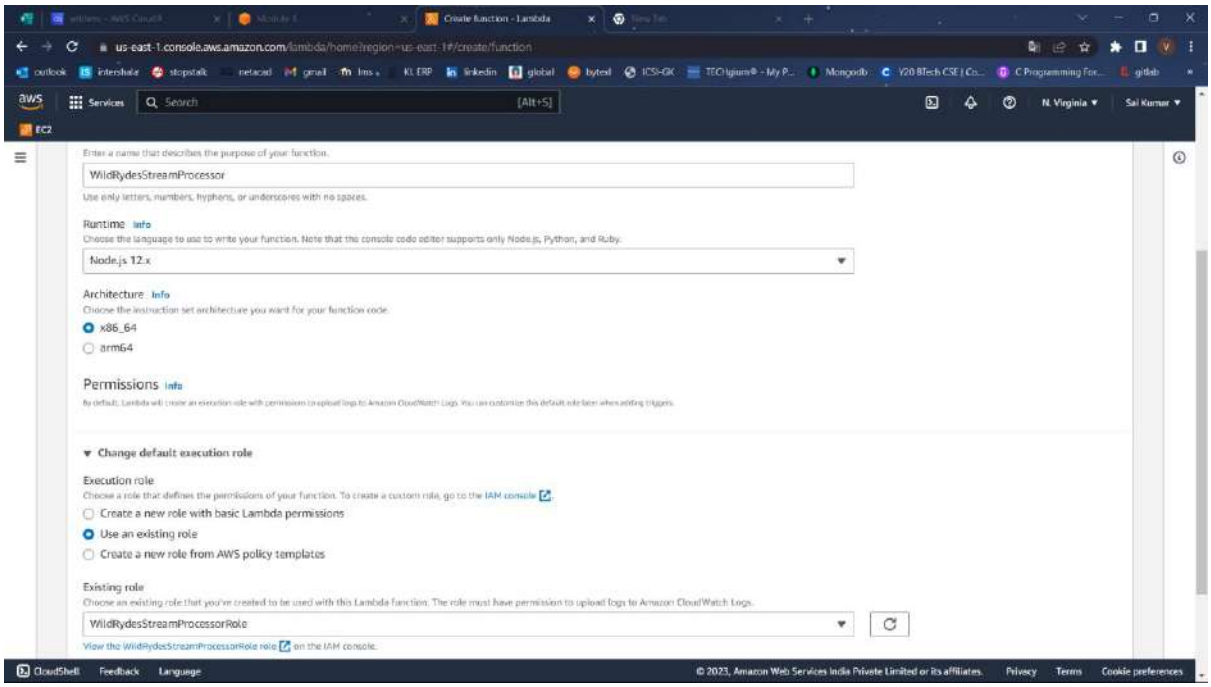

IAM Management Console
us-east-1.console.aws.amazon.com/iamv2/home#/roles/create?commonUseCase=Lambda&step=addPermission&trustedEntityType=AWS_SERVICE
Services
Search
[Alt+S]
Global
Sai Kumar
IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Add permissions Info
Permissions policies (Selected 1/847) Info
Choose one or more policies to attach to your new role.
Create policy
Filter policies by property or policy name and press enter.
1 match
"WildRydesDynamoDBWritePolicy"
Clear filters
Policy name
Type
Description
WildRydesDynamoDB...
Custom...
Set permissions boundary - optional Info
Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting, but you can use it to delegate permission management to others.
Cancel
Previous
Next
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

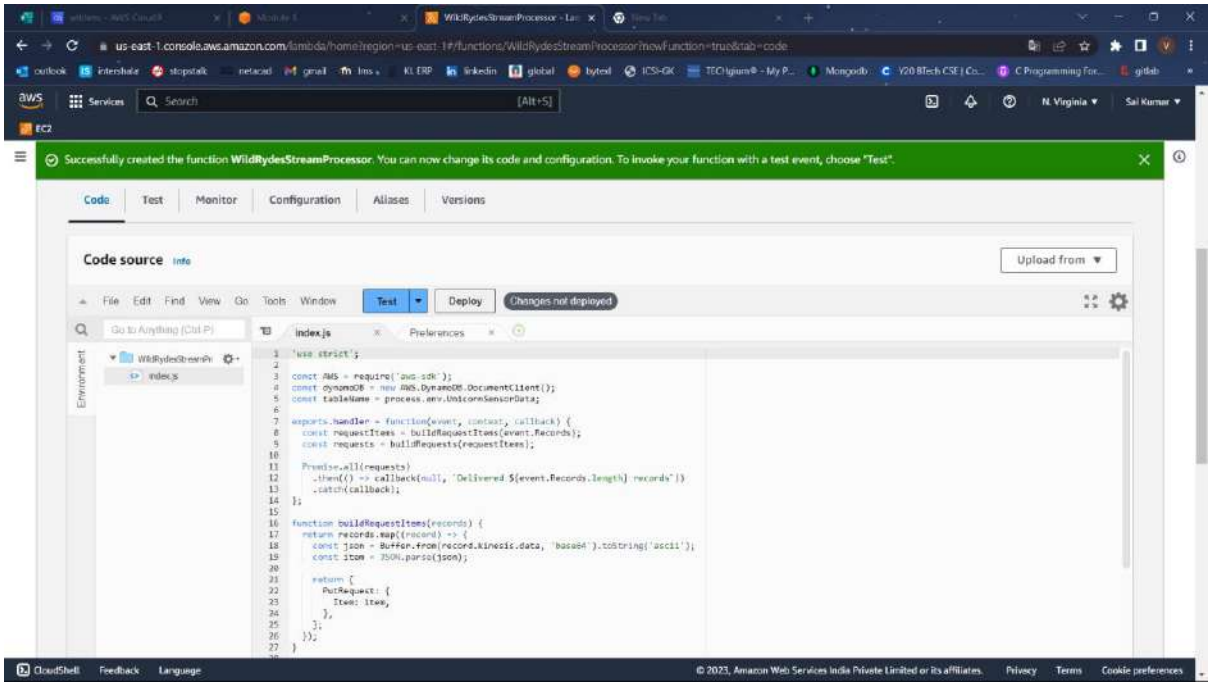

IAM Management Console
us-east-1.console.aws.amazon.com/iamv2/home#/roles/create?commonUseCase=Lambda&policies=arn%3Aaws%3Aiam%3A%3A123805600860%3Apolicy%2FWildRydesDynamoDBWri...
Services
Search
[Alt+S]
Global
Sai Kumar
IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Name, review, and create
Role details
Role name
Enter a meaningful name to identify this role.
WildRydesStreamProcessorRole
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.
Description
Add a short explanation for this role.
Allows Lambda functions to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Step 1: Select trusted entities
Edit
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"sts:AssumeRole"
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences



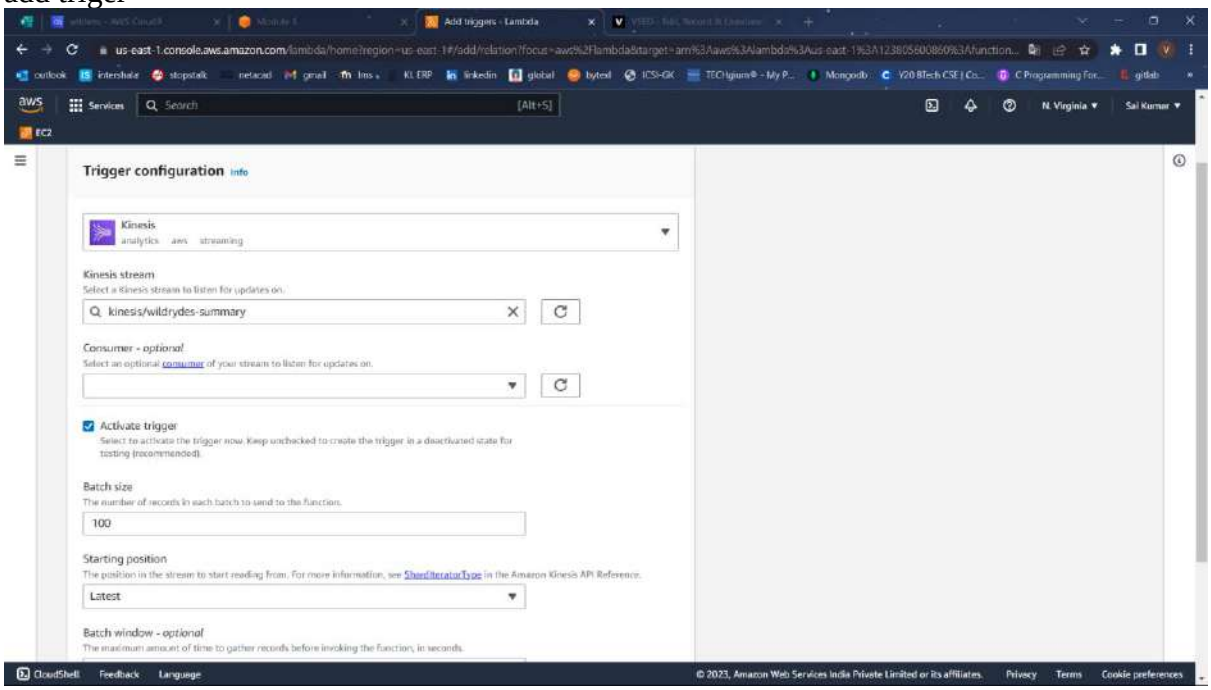
Role created successfully

→ Create a Lambda function to process the stream





add triger

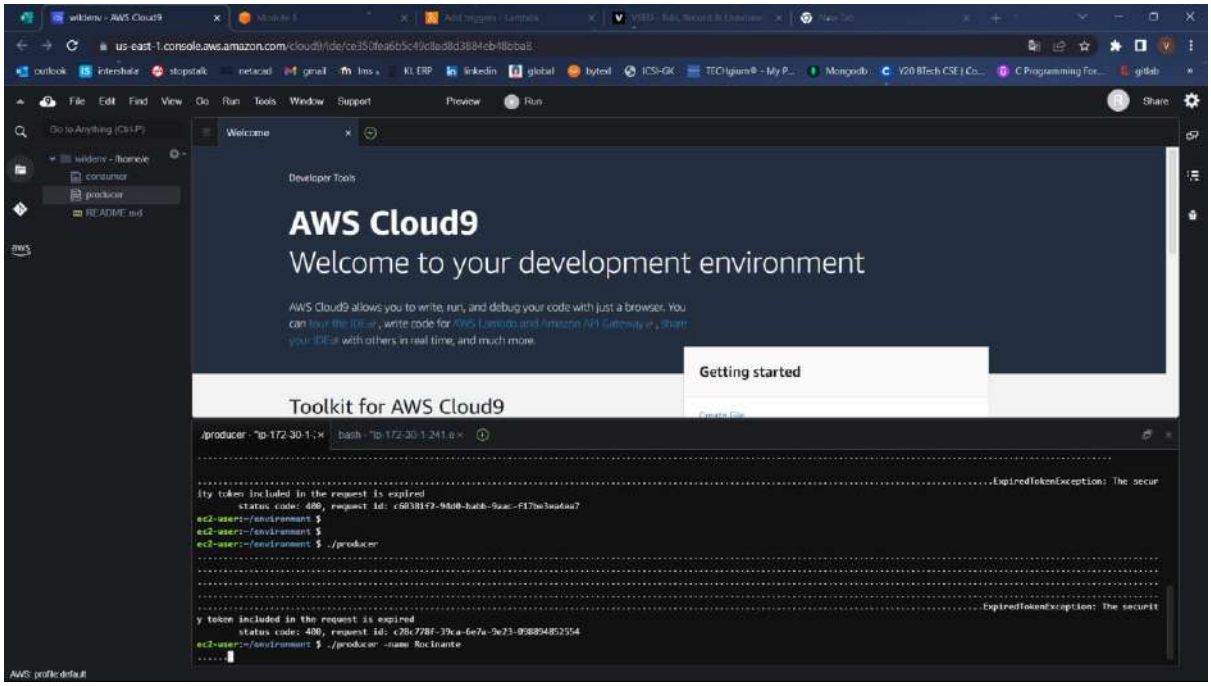


→ Monitor the Lambda function

Run command

/producer -name Rocinante

Return to the *WildRydesStreamProcessor* Lambda function. Select the Monitoring tab and explore the metrics available to monitor the function. Select View Logs in CloudWatch to explore the function's log output



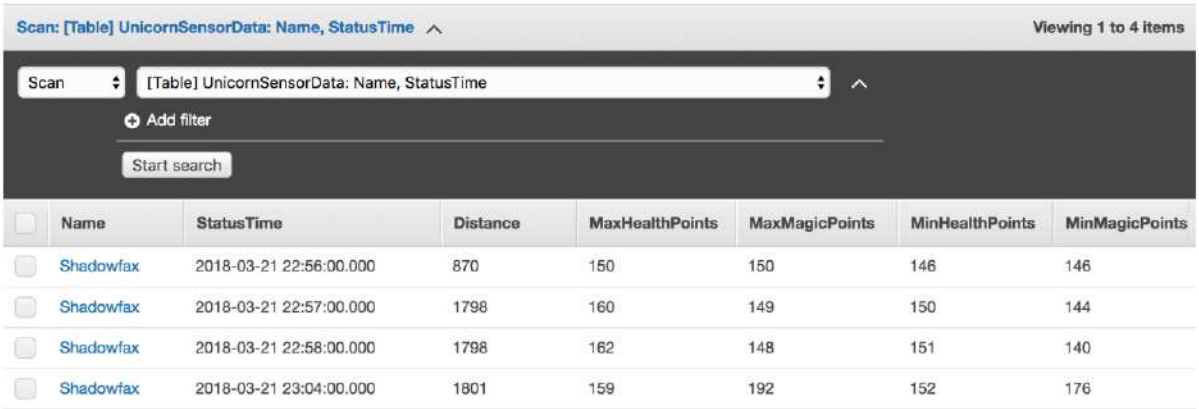
→Query the DynamoDB table

Select Services then select DynamoDB in the Database section.

Select Tables from the left-hand navigation.

Select UnicornSensorData.

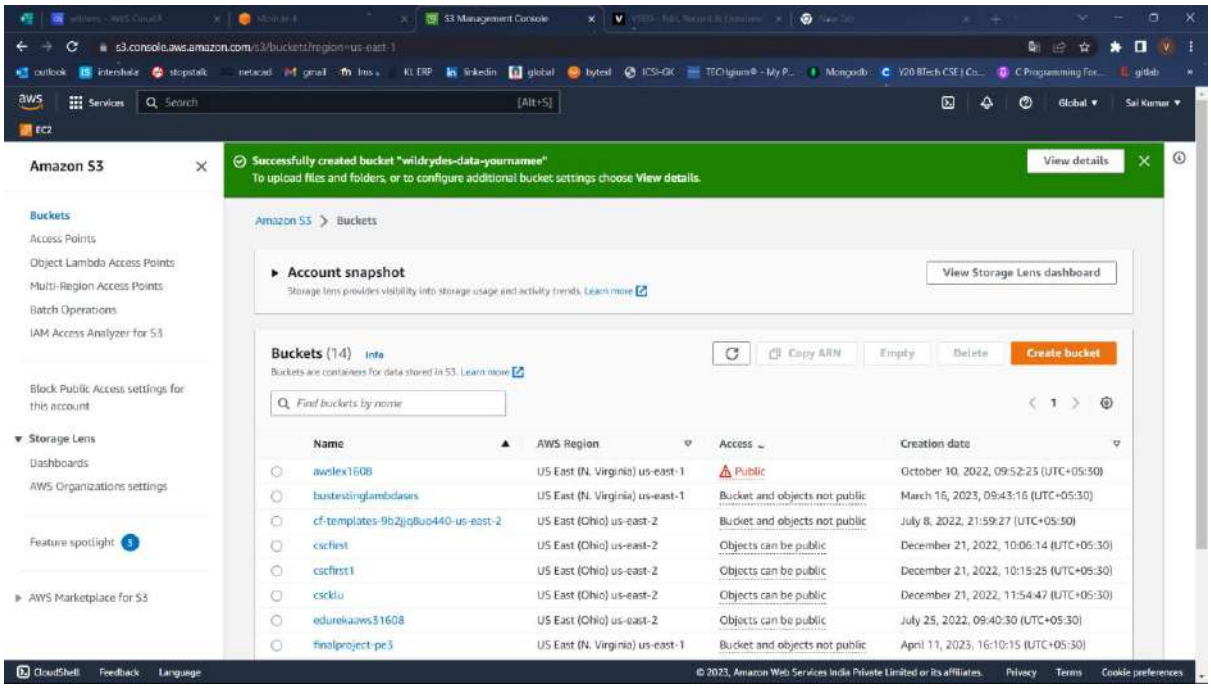
Select the Items tab. Here you should see each per-minute data point for each Unicorn for which you're running a producer.



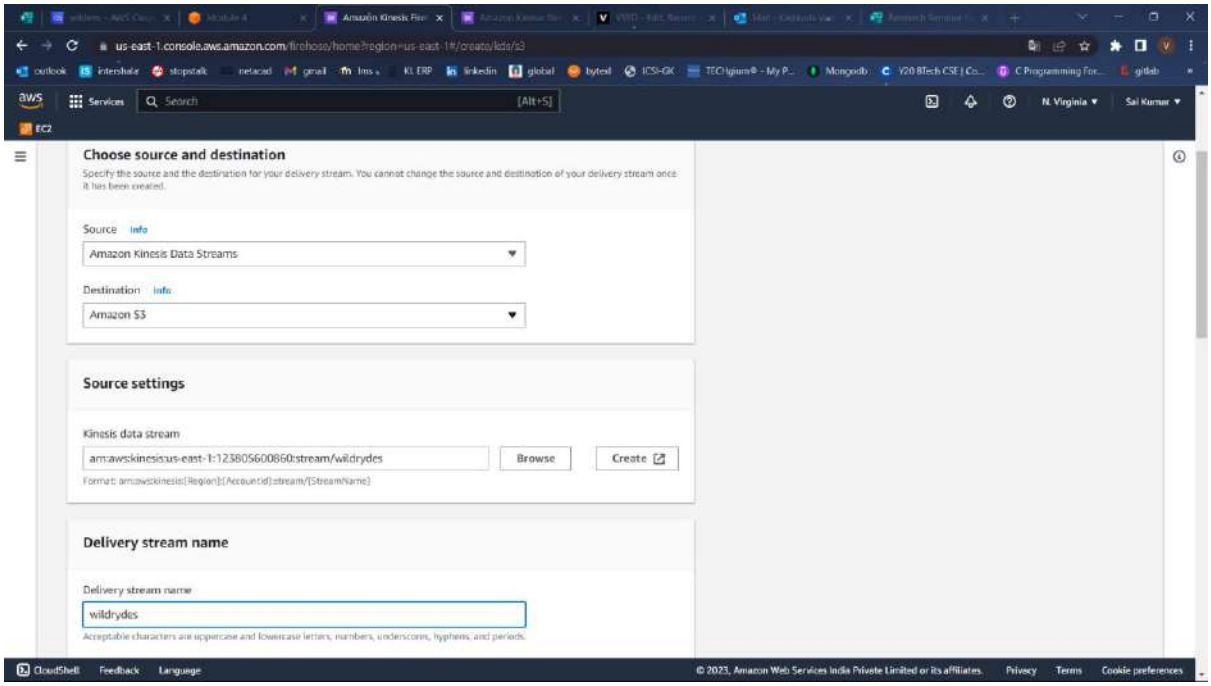
Name	StatusTime	Distance	MaxHealthPoints	MaxMagicPoints	MinHealthPoints	MinMagicPoints
Shadowfax	2018-03-21 22:56:00.000	870	150	150	146	146
Shadowfax	2018-03-21 22:57:00.000	1798	160	149	150	144
Shadowfax	2018-03-21 22:58:00.000	1798	162	148	151	140
Shadowfax	2018-03-21 23:04:00.000	1801	159	192	152	176

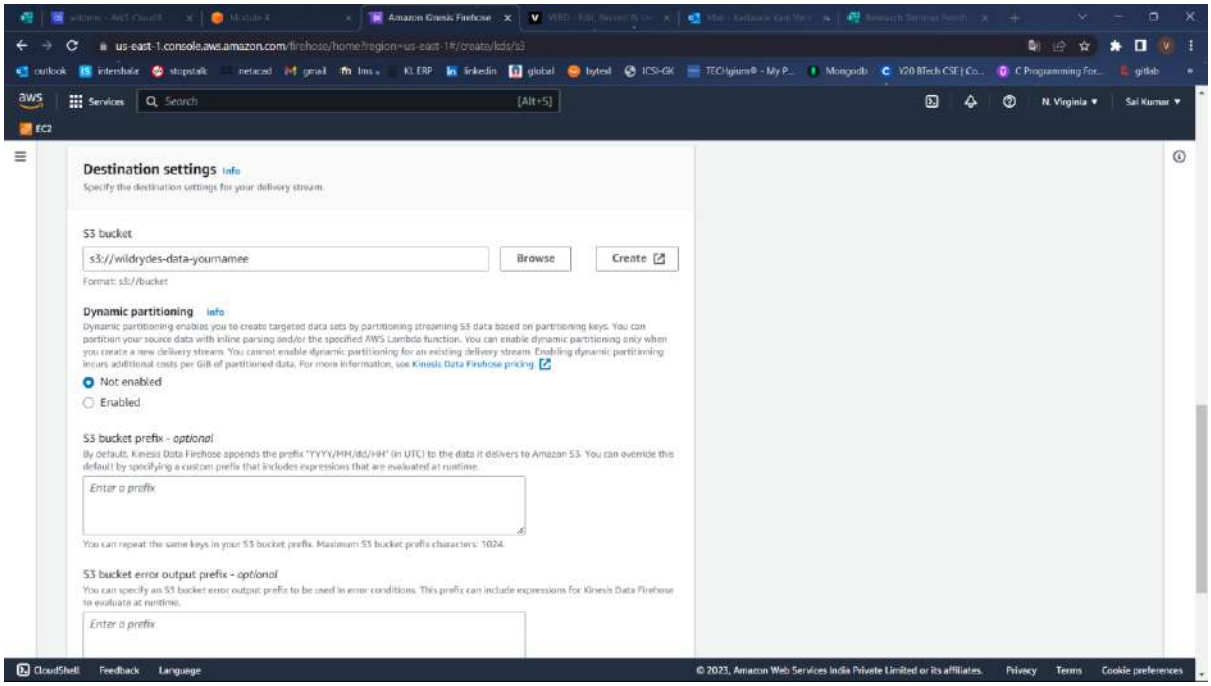
STEP 4 : Store & query Data

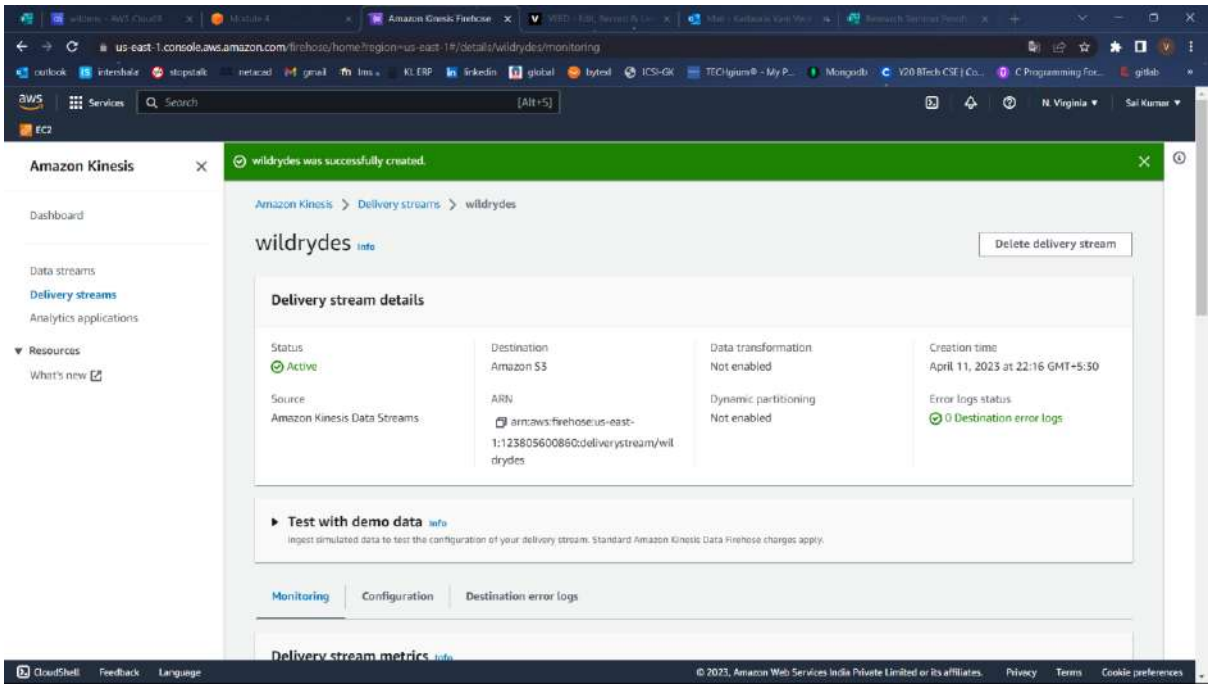
→Create an Amazon S3 bucket



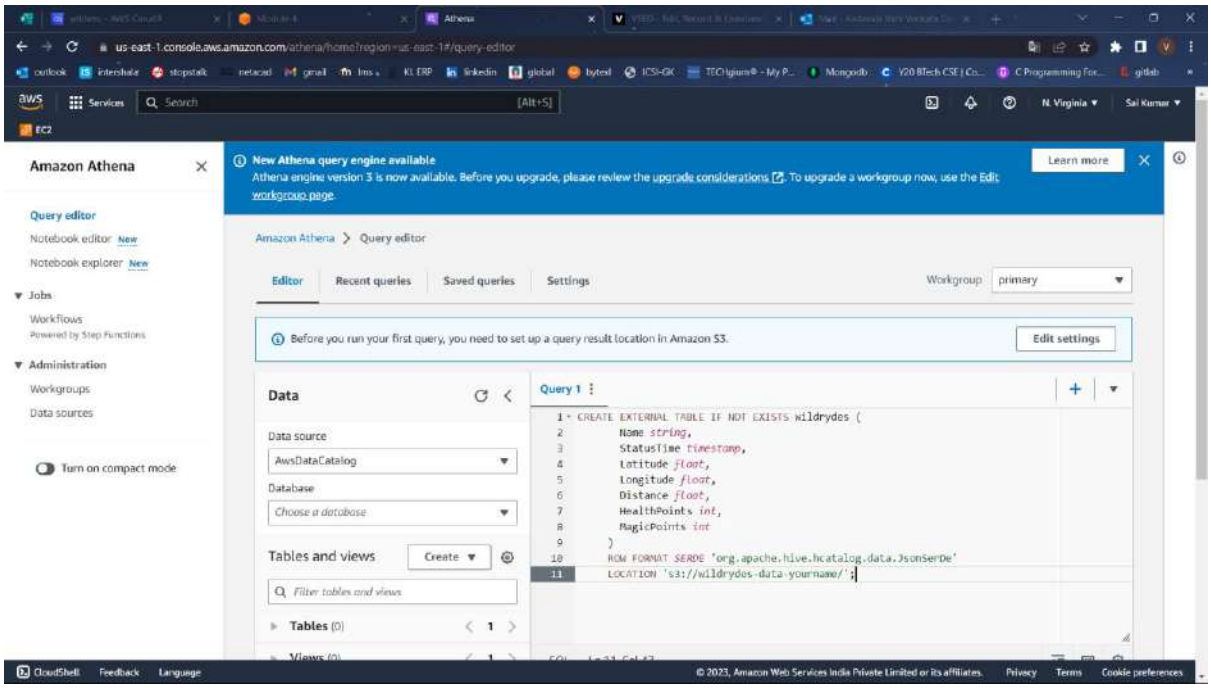
→Create an Amazon Kinesis Data Firehose delivery stream




Destination settings Info
Specify the destination settings for your delivery stream.
S3 bucket
s3://wildrydes-data-yourname
Browse
Create
Format: s3://bucket
Dynamic partitioning Info
Dynamic partitioning enables you to create targeted data sets by partitioning streaming S3 data based on partitioning keys. You can partition your source data with inline parsing and/or the specified AWS Lambda function. You can enable dynamic partitioning only when you create a new delivery stream. You cannot enable dynamic partitioning for an existing delivery stream. Enabling dynamic partitioning incurs additional costs per GiB of partitioned data. For more information, see Kinesis Data Firehose pricing.
Not enabled
Enabled
S3 bucket prefix - optional
By default, Kinesis Data Firehose appends the prefix "YYYY/MM/dd/HH/" (in UTC) to the data it delivers to Amazon S3. You can override this default by specifying a custom prefix that includes expressions that are evaluated at runtime.
Enter a prefix
You can repeat the same keys in your S3 bucket prefix. Maximum S3 bucket prefix characters: 1024.
S3 bucket error output prefix - optional
You can specify an S3 bucket error output prefix to be used in error conditions. This prefix can include expressions for Kinesis Data Firehose to evaluate at runtime.
Enter a prefix
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

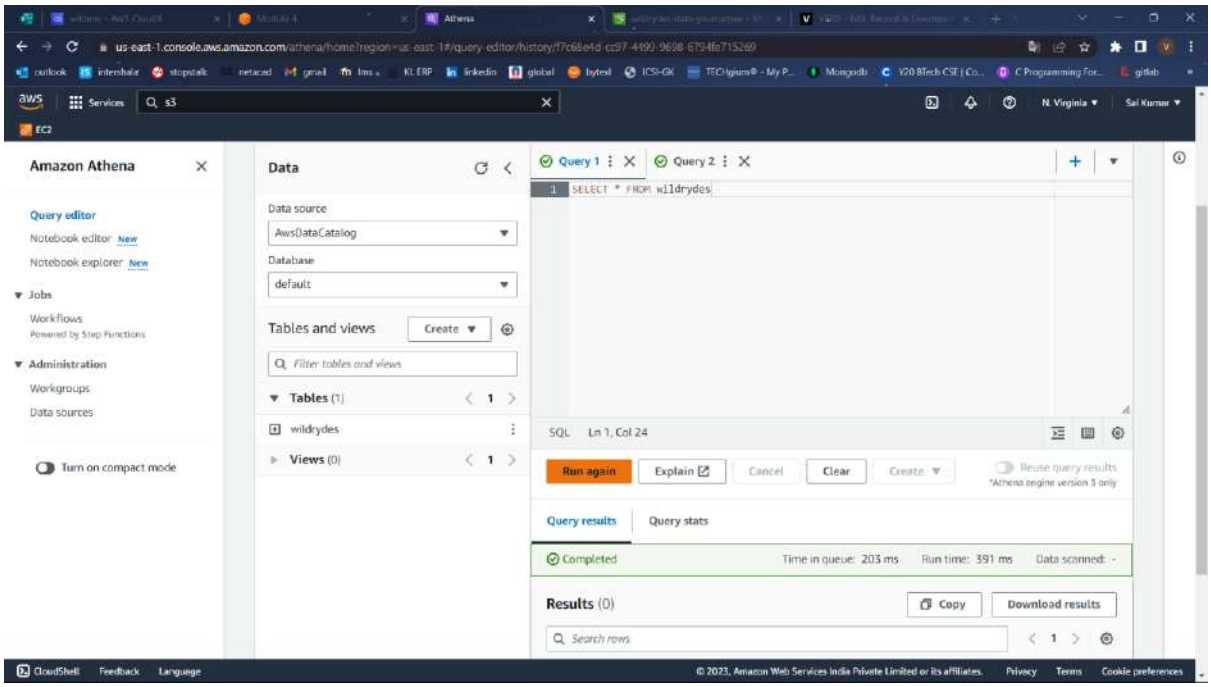

Amazon Kinesis
Dashboard
Data streams
Delivery streams
Analytics applications
Resources
What's new
wildrydes was successfully created.
Amazon Kinesis > Delivery streams > wildrydes
wildrydes Info
Delete delivery stream
Delivery stream details
Status
Active
Destination
Amazon S3
Data transformation
Not enabled
Creation time
April 11, 2023 at 22:16 GMT+5:30
Source
Amazon Kinesis Data Streams
ARN
arn:aws:firehose:us-east-1:123805600860:deliverystream/wildrydes
Dynamic partitioning
Not enabled
Error logs status
0 Destination error logs
Test with demo data Info
Ingest simulated data to test the configuration of your delivery stream. Standard Amazon Kinesis Data Firehose charges apply.
Monitoring
Configuration
Destination error logs
Delivery stream metrics Info
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

→Create an Amazon Athena table



Hence a data table has created , now u can store data by processing directly.

→Query the data files



Hence query runned successfully. Hence we can process data here by storing and query the data.

Therefore , we build a serverless Real-Time Data Processing App using the above services.

STEP 5 : Clean Up all the services created