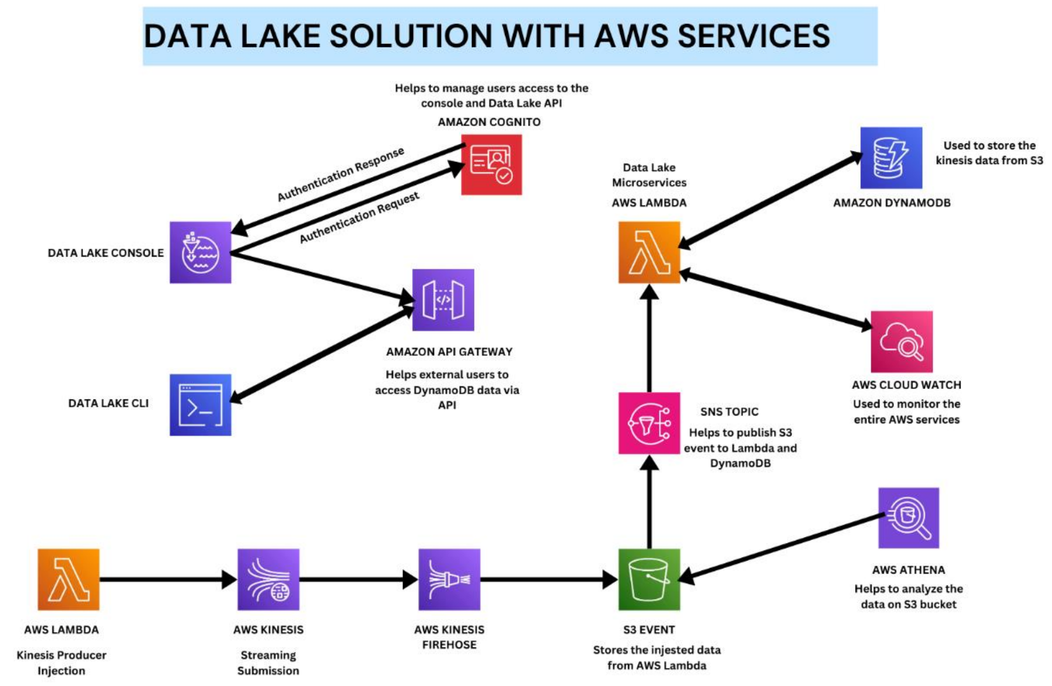
# **CAPSTONE PROJECT**

## **ARCHTECTING AND DEPLOYING A DATA LAKE AUTOMATION SOLUTION**


DATA LAKE SOLUTION WITH AWS SERVICES
Helps to manage users access to the console and Data Lake API
AMAZON COGNITO
Authentication Response
Authentication Request
DATA LAKE CONSOLE
AMAZON API GATEWAY
Helps external users to access DynamoDB data via API
DATA LAKE CLI
Data Lake Microservices
AWS LAMBDA
Used to store the kinesis data from S3
AMAZON DYNAMODB
AWS CLOUD WATCH
Used to monitor the entire AWS services
SNS TOPIC
Helps to publish S3 event to Lambda and DynamoDB
AWS ATHENA
Helps to analyze the data on S3 bucket
AWS LAMBDA
Kinesis Producer Injection
AWS KINESIS
Streaming Submission
AWS KINESIS FIREHOSE
S3 EVENT
Stores the injested data from AWS Lambda

## **PROJECT DESCRIPTION**

This Data Lake project is designed to help Datasoft Incorporation solve the data needs of its large pool of clients by providing a more streamlined process and platform to enable the company to easily correlate, transform, query, analyze and visualize customers data feeds at every point in time making it easy to generate valuable insights necessary in meeting customers' needs as well as improving the revenue of the company.

## **AWS SERVICE TOOLS USED FOR THIS PROJECT**

- AWS Lambda
- Amazon Kinesis Data Stream
- Amazon Kinesis Firehose
- Amazon S3
- Amazon Athena
- Amazon SNS TOPIC
- Amazon DynamoDB
- Amazon CloudWatch
- AWS Cognito
- Amazon API Gateway

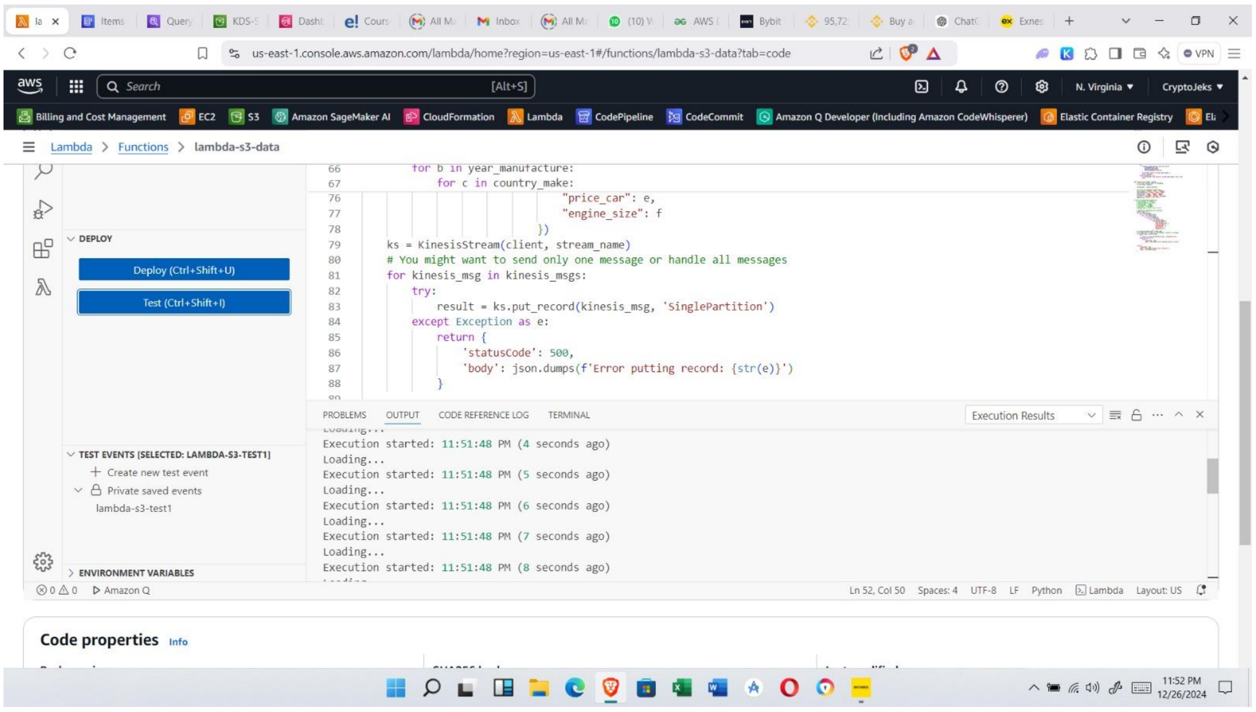
## PROJECT SECTIONS

This project is divided into three major sections which includes,

- Data Ingestion Section
- Data Processing and Storage Section
- API Gateway Section

## DATA INJESTION SECTION

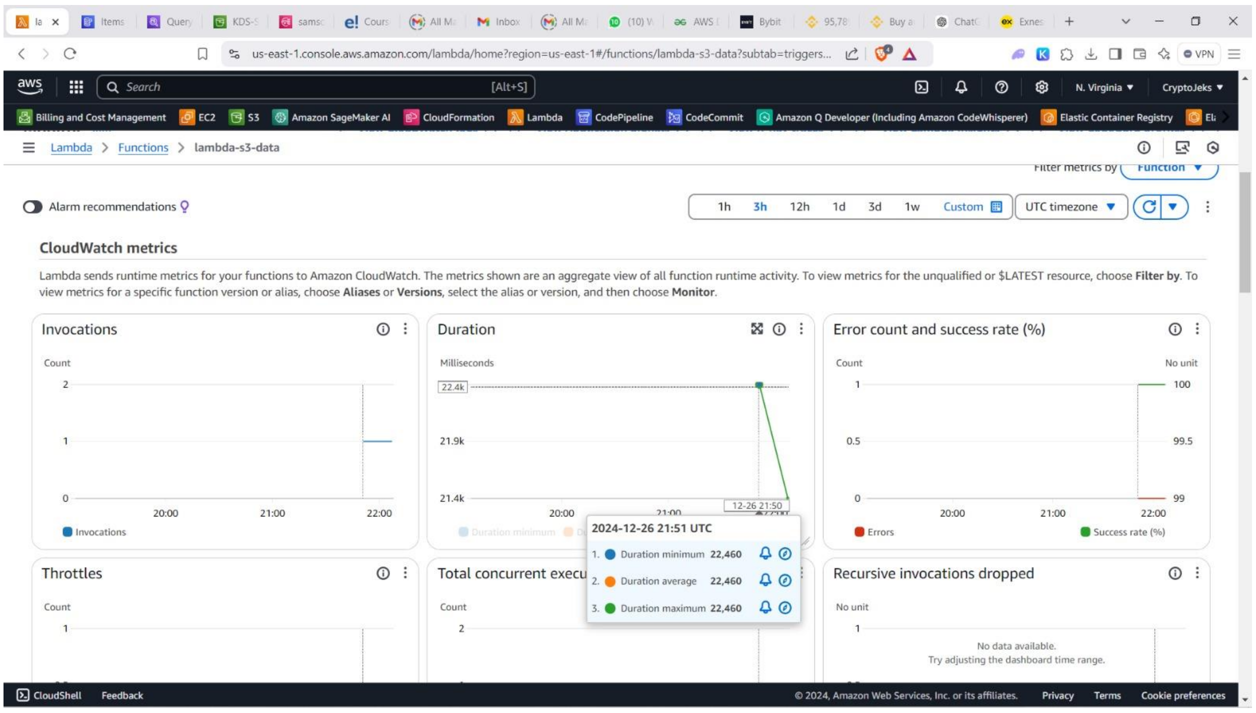
This section of the project comprises of AWS services such as AWS lambda, Amazon Kinesis Data Stream, Amazon Kinesis Firehose, and Amazon S3 Bucket. In this section, I ingested a Kinesis Producer Python Code which contains a kinesis message about vehicle information using an AWS lambda function. Kindly click on this link to access my code on Github, https://github.com/CryptoJeks/capstone-code. Using IAM, I assigned a role for the lambda function with full permission to access Amazon Kinesis Data Stream, Kinesis Firehose, and Amazon S3 bucket. This made it possible for my Kinesis Producer Python Code to send the kinesis message to the s3 Bucket via Kinesis Data Stream and Kinesis Firehose. Find below some screenshots of my code and also the data ingestion process.



```
        for b in year_manufacture:
            for c in country_make:
                                "price_car": e,
                                "engine_size": f
                            })
    ks = KinesisStream(client, stream_name)
    # You might want to send only one message or handle all messages
    for kinesis_msg in kinesis_msgs:
        try:
            result = ks.put_record(kinesis_msg, 'SinglePartition')
        except Exception as e:
            return {
                'statusCode': 500,
                'body': json.dumps(f'Error putting record: {str(e)}')
            }
```

```
Execution started: 11:51:48 PM (4 seconds ago)
Loading...
Execution started: 11:51:48 PM (5 seconds ago)
Loading...
Execution started: 11:51:48 PM (6 seconds ago)
Loading...
Execution started: 11:51:48 PM (7 seconds ago)
Loading...
Execution started: 11:51:48 PM (8 seconds ago)
```

**The Kinesis Producer Python Code**



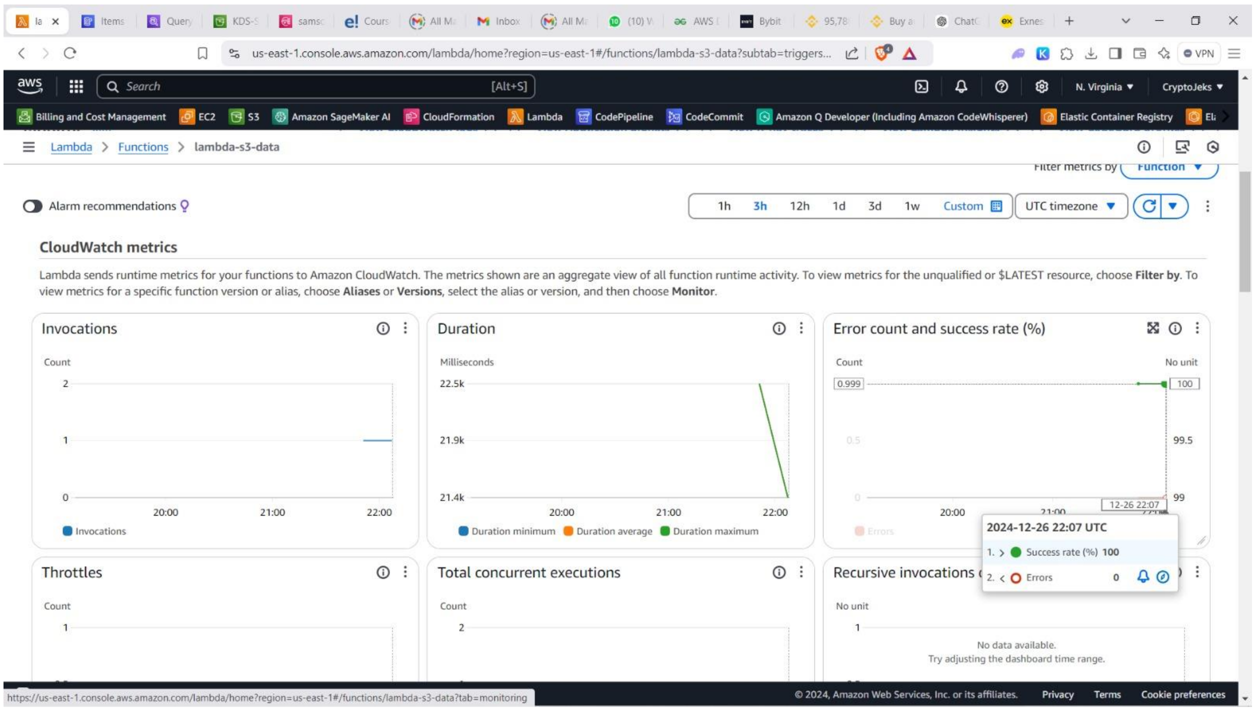
us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/lambda-s3-data?subtab=triggers...

**CloudWatch metrics**

Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an aggregate view of all function runtime activity. To view metrics for the unqualified or $LATEST resource, choose **Filter by**. To view metrics for a specific function version or alias, choose **Aliases** or **Versions**, select the alias or version, and then choose **Monitor**.

No data available.
Try adjusting the dashboard time range.

© 2024, Amazon Web Services, Inc. or its affiliates.


Lambda > Functions > lambda-s3-data
Alarm recommendations
1h 3h 12h 1d 3d 1w Custom
UTC timezone
Invocations
Duration
Error count and success rate (%)
Throttles
Total concurrent executions
Recursive invocations dropped
2024-12-26 22:07 UTC
Success rate (%) 100
Errors 0

**CloudWatch metrics**

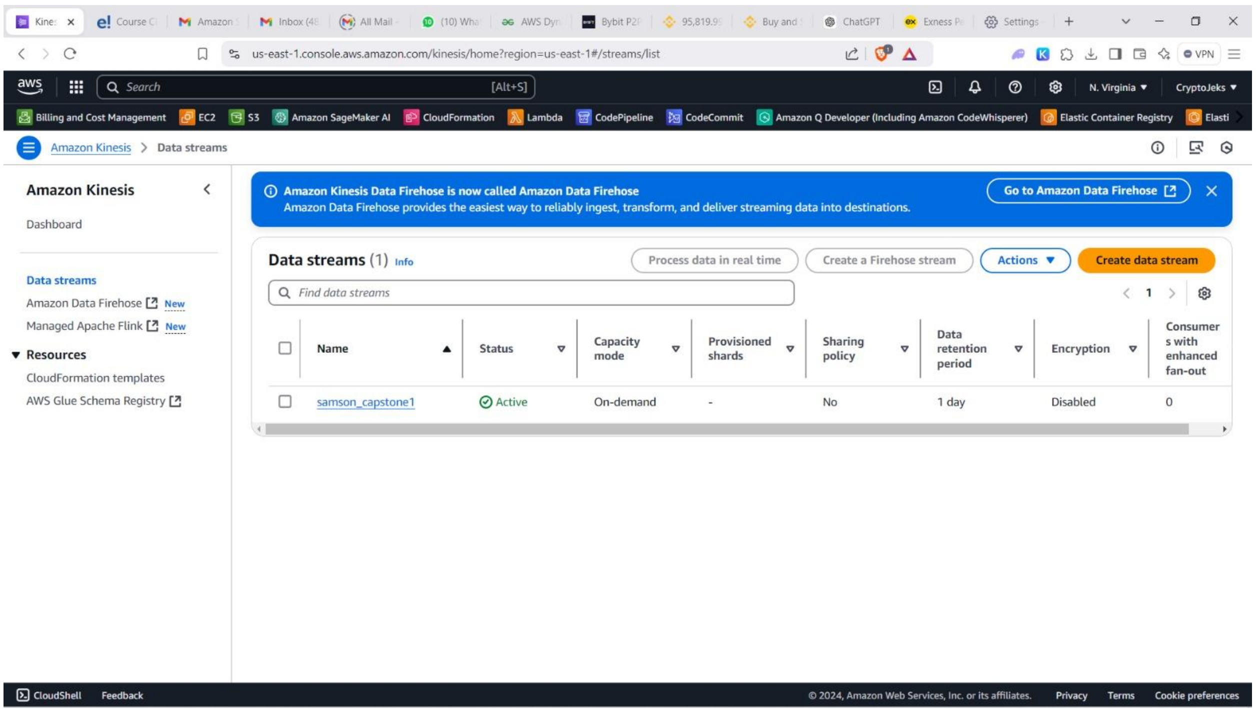
Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an aggregate view of all function runtime activity. To view metrics for the unqualified or $LATEST resource, choose **Filter by**. To view metrics for a specific function version or alias, choose **Aliases** or **Versions**, select the alias or version, and then choose **Monitor**.

No data available.
Try adjusting the dashboard time range.

https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/lambda-s3-data?tab=monitoring

© 2024, Amazon Web Services, Inc. or its affiliates.

**Cloud Watch showing the activities of Kinesis Producer**

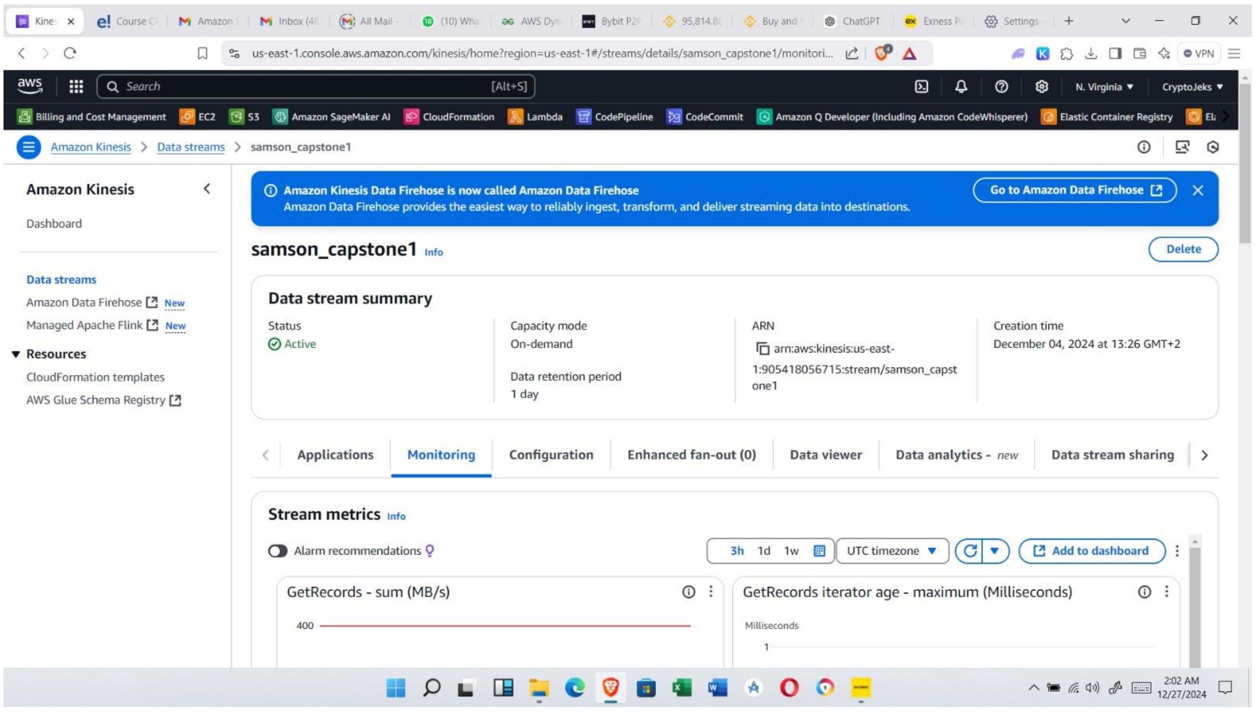


**Amazon Kinesis Data Firehose is now called Amazon Data Firehose**
Amazon Data Firehose provides the easiest way to reliably ingest, transform, and deliver streaming data into destinations.

Data streams (1) Info

| Name | Status | Capacity mode | Provisioned shards | Sharing policy | Data retention period | Encryption | Consumers with enhanced fan-out |
|---|---|---|---|---|---|---|---|
| samson_capstone1 | Active | On-demand | - | No | 1 day | Disabled | 0 |

## My Amazon Kinesis Data Stream service



**Amazon Kinesis Data Firehose is now called Amazon Data Firehose**
Amazon Data Firehose provides the easiest way to reliably ingest, transform, and deliver streaming data into destinations.

samson_capstone1 Info

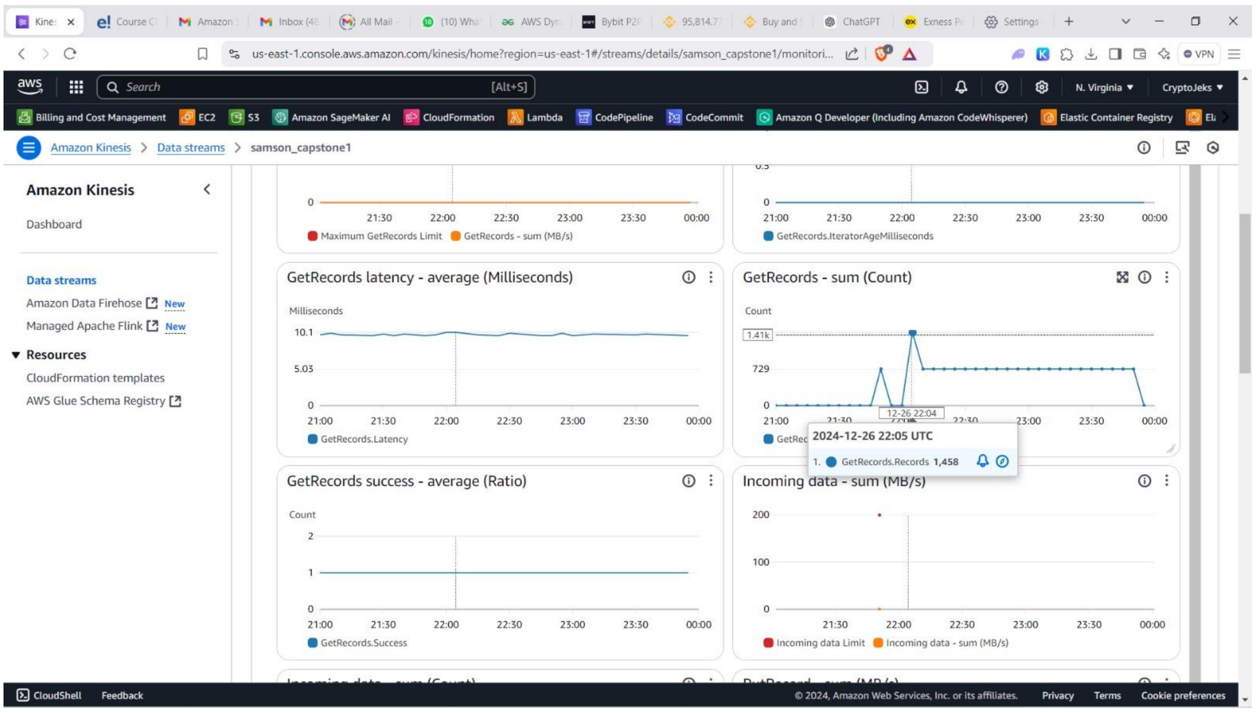
Data stream summary

Status: Active
Capacity mode: On-demand
Data retention period: 1 day
ARN: arn:aws:kinesis:us-east-1:905418056715:stream/samson_capstone1
Creation time: December 04, 2024 at 13:26 GMT+2

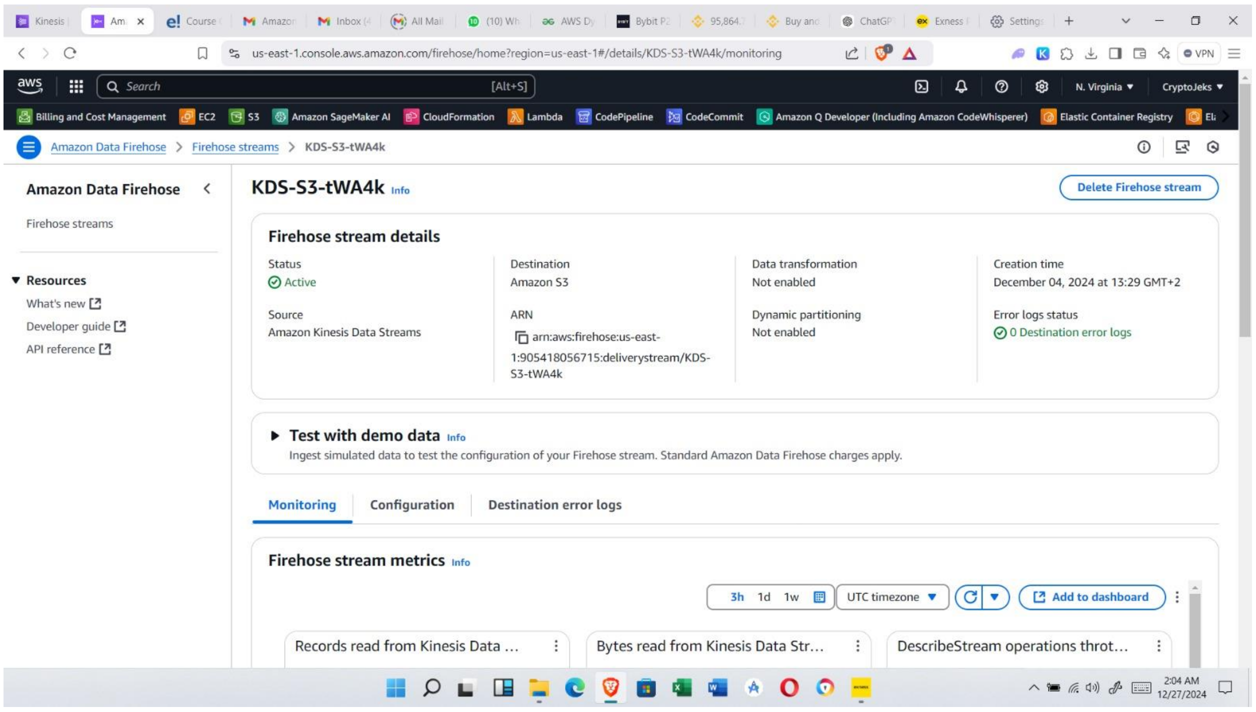
Stream metrics Info

GetRecords - sum (MB/s)
400

GetRecords iterator age - maximum (Milliseconds)
Milliseconds
1



**Real-Time Streaming of Kinesis Message via Kinesis Data Stream**



KDS-S3-tWA4k Info

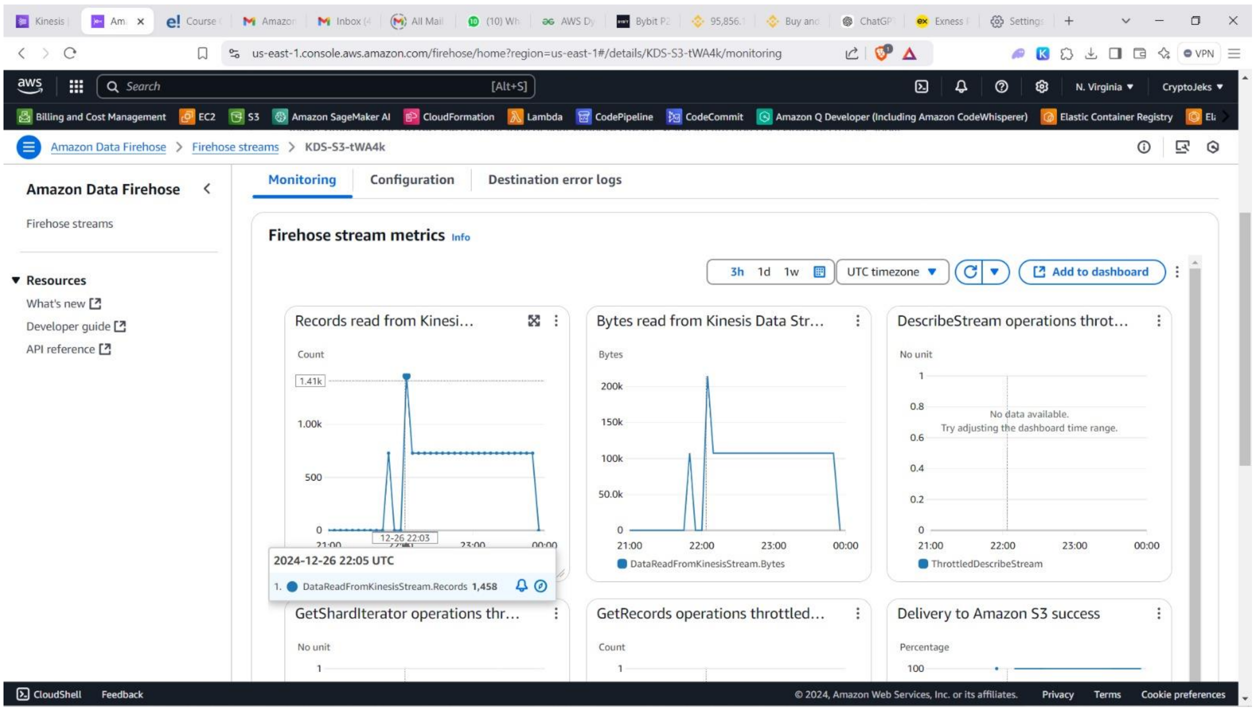
Firehose stream details

| Status | Destination | Data transformation | Creation time |
|---|---|---|---|
| Active | Amazon S3 | Not enabled | December 04, 2024 at 13:29 GMT+2 |
| Source | ARN | Dynamic partitioning | Error logs status |
| Amazon Kinesis Data Streams | arn:aws:firehose:us-east-1:905418056715:deliverystream/KDS-S3-tWA4k | Not enabled | 0 Destination error logs |

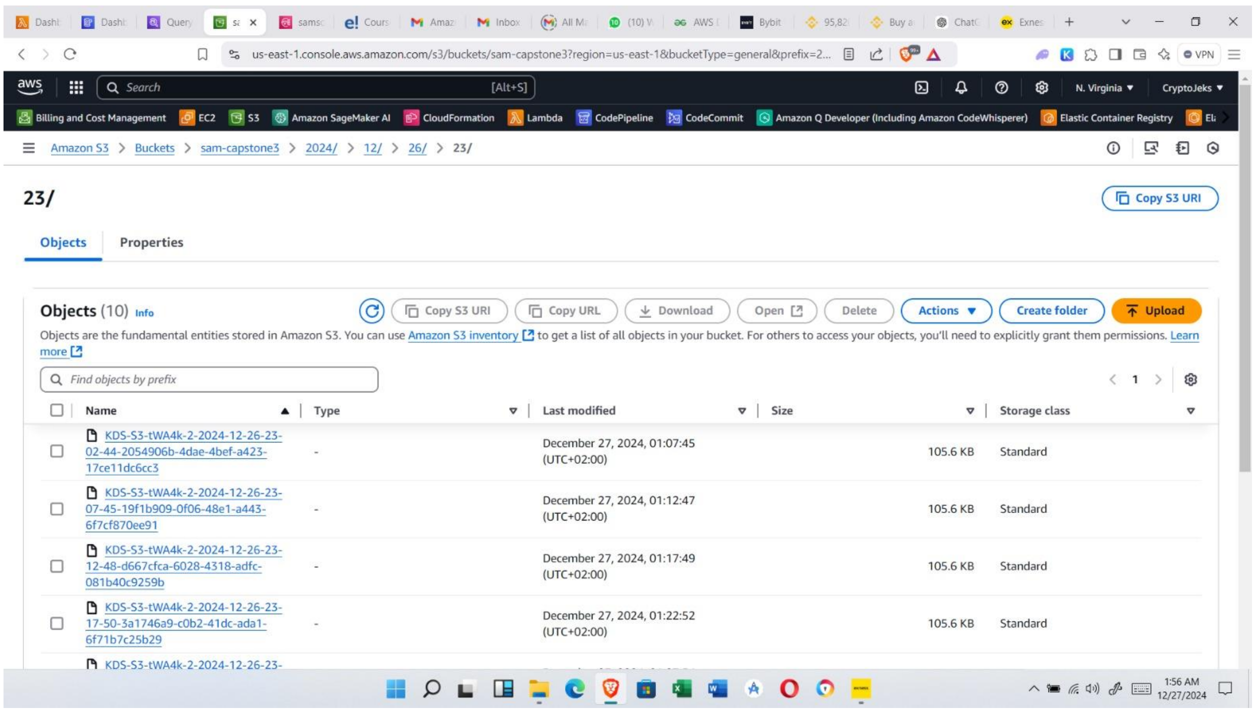
Test with demo data Info
Ingest simulated data to test the configuration of your Firehose stream. Standard Amazon Data Firehose charges apply.

Amazon Data Firehose > Firehose streams > KDS-S3-tWA4k
Delete Firehose stream
Monitoring
Configuration
Destination error logs
Firehose stream metrics Info
3h 1d 1w
UTC timezone
Add to dashboard
Records read from Kinesis Data ...
Bytes read from Kinesis Data Str...
DescribeStream operations throt...

**My Amazon Kinesis Firehose Stream service**


us-east-1.console.aws.amazon.com/firehose/home?region=us-east-1#/details/KDS-S3-tWA4k/monitoring
Amazon Data Firehose > Firehose streams > KDS-S3-tWA4k
Monitoring
Configuration
Destination error logs
Firehose stream metrics
3h 1d 1w
UTC timezone
Add to dashboard
Records read from Kinesi…
Bytes read from Kinesis Data Str…
DescribeStream operations throt…
No data available.
Try adjusting the dashboard time range.
2024-12-26 22:05 UTC
1. DataReadFromKinesisStream.Records 1,458
DataReadFromKinesisStream.Bytes
ThrottledDescribeStream
GetShardIterator operations thr…
GetRecords operations throttled…
Delivery to Amazon S3 success

**Live Streaming of Kinesis Message via Amazon Kinesis Firehose**



us-east-1.console.aws.amazon.com/s3/buckets/sam-capstone3?region=us-east-1&bucketType=general&prefix=2...

Amazon S3 > Buckets > sam-capstone3 > 2024/ > 12/ > 26/ > 23/

23/

Objects (10)

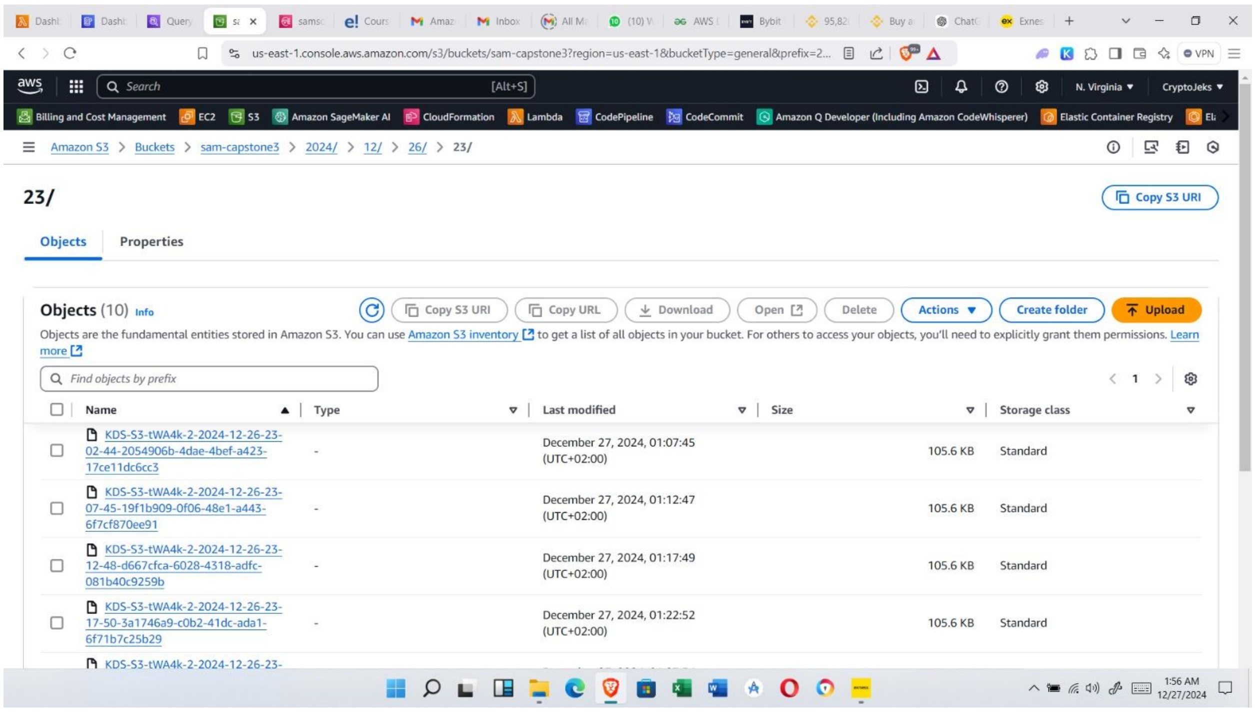
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

| Name | Type | Last modified | Size | Storage class |
|---|---|---|---|---|
| KDS-S3-tWA4k-2-2024-12-26-23-02-44-2054906b-4dae-4bef-a423-17ce11dc6cc3 | - | December 27, 2024, 01:07:45 (UTC+02:00) | 105.6 KB | Standard |
| KDS-S3-tWA4k-2-2024-12-26-23-07-45-19f1b909-0f06-48e1-a443-6f7cf870ee91 | - | December 27, 2024, 01:12:47 (UTC+02:00) | 105.6 KB | Standard |
| KDS-S3-tWA4k-2-2024-12-26-23-12-48-d667cfca-6028-4318-adfc-081b40c9259b | - | December 27, 2024, 01:17:49 (UTC+02:00) | 105.6 KB | Standard |
| KDS-S3-tWA4k-2-2024-12-26-23-17-50-3a1746a9-c0b2-41dc-ada1-6f71b7c25b29 | - | December 27, 2024, 01:22:52 (UTC+02:00) | 105.6 KB | Standard |
| KDS-S3-tWA4k-2-2024-12-26-23- | | | | |

**Kinesis Message Successfully pushed to my S3 bucket using a lambda function via kinesis data stream and Firehose.**

## DATA PROCESSING AND STORTAGE SECTION

The data processing section comprises of AWS services such as the S3 Bucket, SNS Topic, AWS Lambda, Amazon DynamoDB and Amazon Athena. In this section, the kinesis message stored in the S3 bucket in JASON format triggers an SNS Topic as soon as there is a PUT into the S3 bucket. SNS Topic then publishes the JASON Kinesis Message into DynamoDB in a visible structural form with the help of a lambda function. Once the kinesis message is published on DynamoDB, Amazon Athena is used to analyze and generate insight from the published Data on DynamoDB. Below is the screenshot of my published kinesis message on my S3 Bucket, SNS Topic and the kinesis message on my DynamoDB.


us-east-1.console.aws.amazon.com/s3/buckets/sam-capstone3?region=us-east-1&bucketType=general&prefix=2...
Amazon S3 > Buckets > sam-capstone3 > 2024/ > 12/ > 26/ > 23/
Copy S3 URI
Objects
Properties
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
1:56 AM
12/27/2024

23/

Objects (10) Info

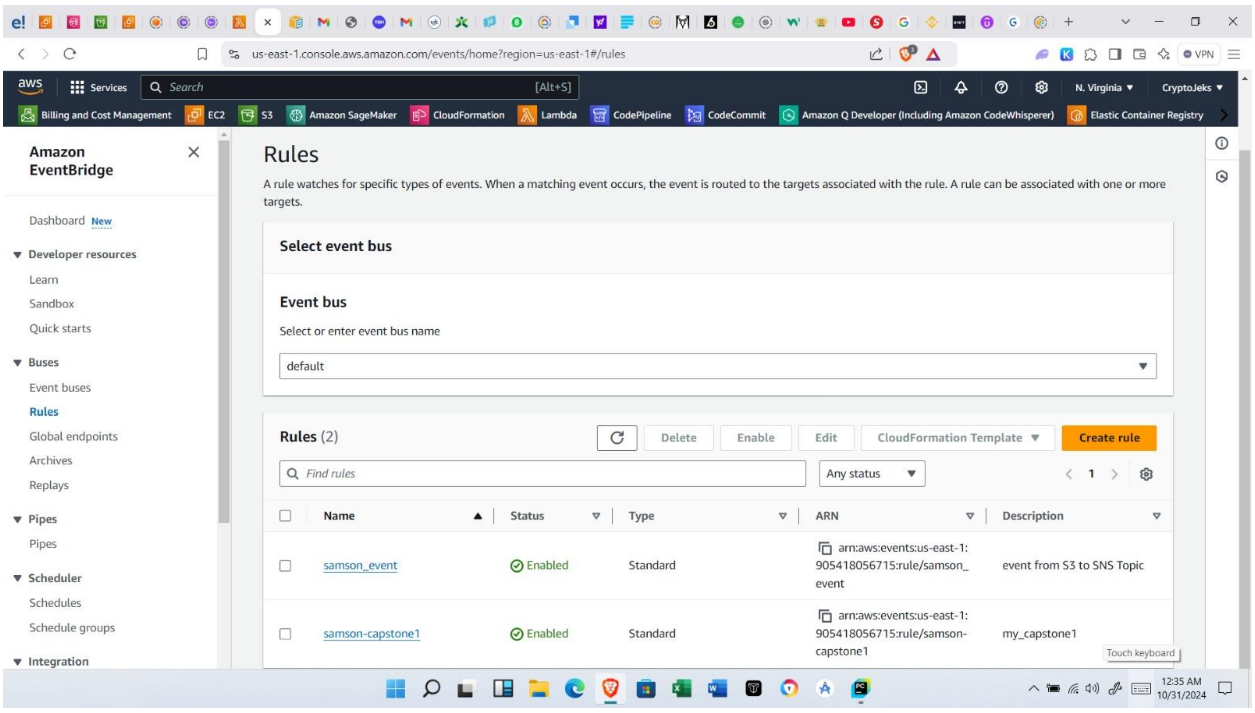
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

| Name | Type | Last modified | Size | Storage class |
|---|---|---|---|---|
| KDS-S3-tWA4k-2-2024-12-26-23-02-44-2054906b-4dae-4bef-a423-17ce11dc6cc3 | - | December 27, 2024, 01:07:45 (UTC+02:00) | 105.6 KB | Standard |
| KDS-S3-tWA4k-2-2024-12-26-23-07-45-19f1b909-0f06-48e1-a443-6f7cf870ee91 | - | December 27, 2024, 01:12:47 (UTC+02:00) | 105.6 KB | Standard |
| KDS-S3-tWA4k-2-2024-12-26-23-12-48-d667cfca-6028-4318-adfc-081b40c9259b | - | December 27, 2024, 01:17:49 (UTC+02:00) | 105.6 KB | Standard |
| KDS-S3-tWA4k-2-2024-12-26-23-17-50-3a1746a9-c0b2-41dc-ada1-6f71b7c25b29 | - | December 27, 2024, 01:22:52 (UTC+02:00) | 105.6 KB | Standard |
| KDS-S3-tWA4k-2-2024-12-26-23- | | | | |

**Kinesis Message pushed to S3 via AWS Lambda**

## Amazon Event Bridge Creation

Amazon Event Bridge is a serverless service that uses events to connect application components together, making it easier for developers to build scalable event-driven applications. In this case, the Event Bridge is connecting my S3 bucket to DynamoDB by publishing my kinesis message PUT on S3 to DynamoDB using an SNS Topic. Find below some of my live screenshots explaining the process.



Rules

A rule watches for specific types of events. When a matching event occurs, the event is routed to the targets associated with the rule. A rule can be associated with one or more targets.

Select event bus

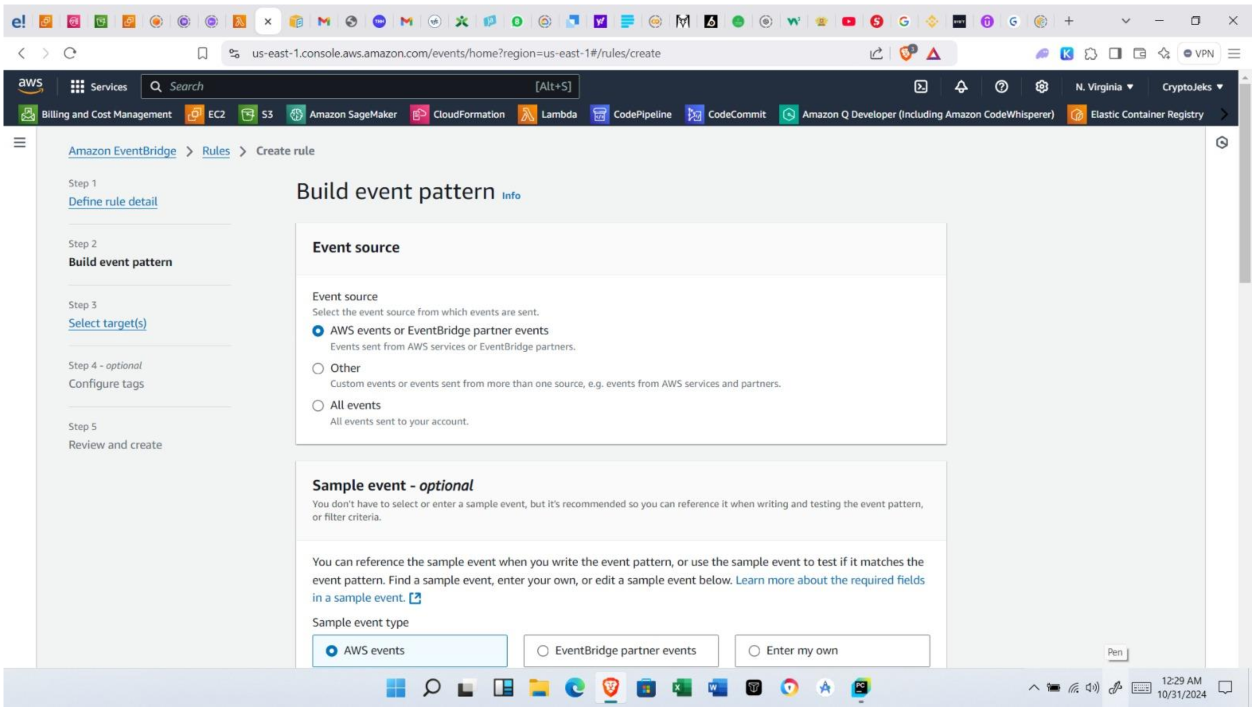
Event bus

Select or enter event bus name

default

Rules (2)

| Name | Status | Type | ARN | Description |
|---|---|---|---|---|
| samson_event | Enabled | Standard | arn:aws:events:us-east-1:905418056715:rule/samson_event | event from S3 to SNS Topic |
| samson-capstone1 | Enabled | Standard | arn:aws:events:us-east-1:905418056715:rule/samson-capstone1 | my_capstone1 |



Build event pattern Info

Event source

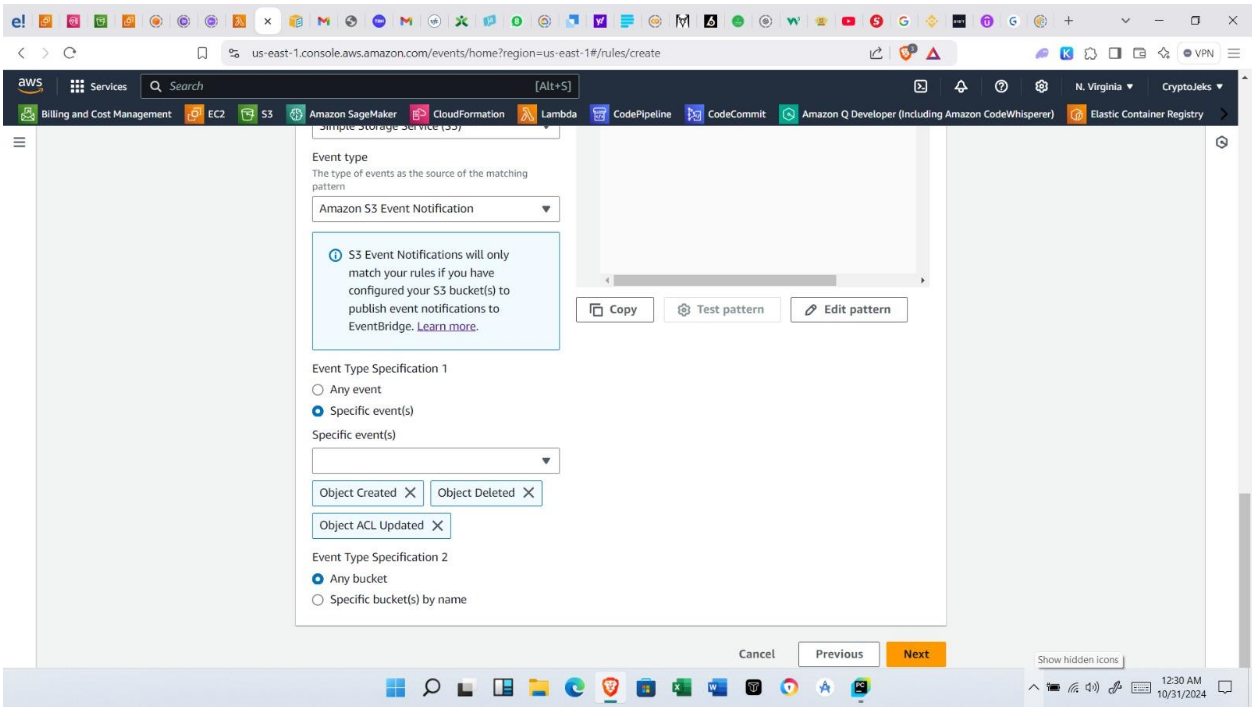
Event source
Select the event source from which events are sent.
- [✓ AWS events or EventBridge partner events] Events sent from AWS services or EventBridge partners.
- Other: Custom events or events sent from more than one source, e.g. events from AWS services and partners.
- All events: All events sent to your account.

Sample event - *optional*

You don't have to select or enter a sample event, but it's recommended so you can reference it when writing and testing the event pattern, or filter criteria.

You can reference the sample event when you write the event pattern, or use the sample event to test if it matches the event pattern. Find a sample event, enter your own, or edit a sample event below. Learn more about the required fields in a sample event.

Sample event type: [✓ AWS events] EventBridge partner events / Enter my own

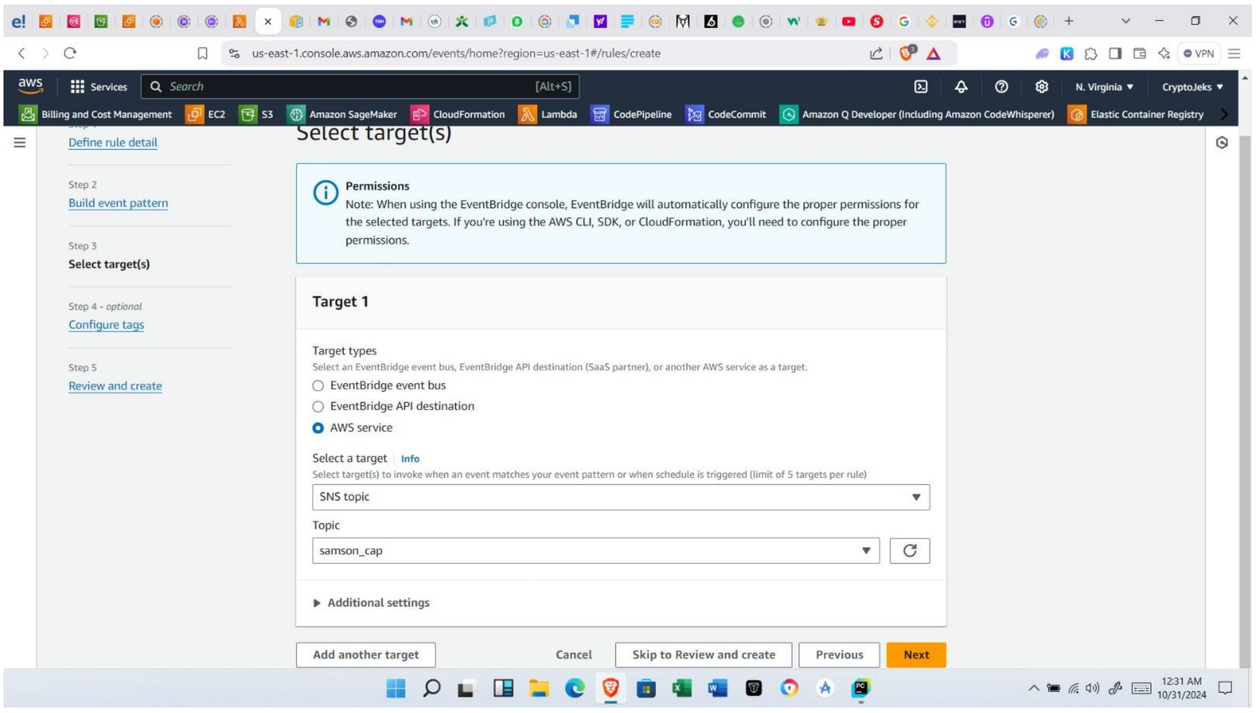

Event type
The type of events as the source of the matching pattern
Amazon S3 Event Notification

S3 Event Notifications will only match your rules if you have configured your S3 bucket(s) to publish event notifications to EventBridge. Learn more.

Event Type Specification 1
( ) Any event
(•) Specific event(s)

Specific event(s)
Object Created ✕ Object Deleted ✕ Object ACL Updated ✕

Event Type Specification 2
(•) Any bucket
( ) Specific bucket(s) by name


Select target(s)

**Permissions**
Note: When using the EventBridge console, EventBridge will automatically configure the proper permissions for the selected targets. If you're using the AWS CLI, SDK, or CloudFormation, you'll need to configure the proper permissions.

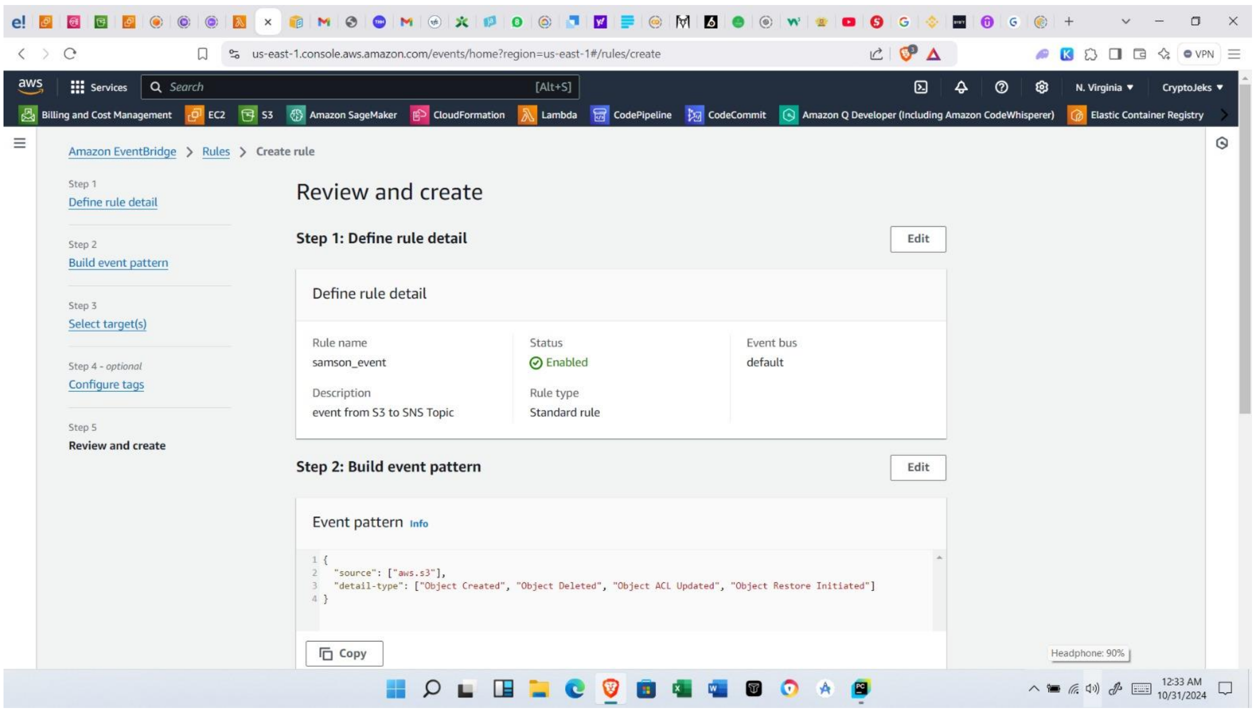
Step 2 Build event pattern
Step 3 Select target(s)
Step 4 - optional Configure tags
Step 5 Review and create

**Target 1**

Target types
Select an EventBridge event bus, EventBridge API destination (SaaS partner), or another AWS service as a target.
( ) EventBridge event bus
( ) EventBridge API destination
(•) AWS service

Select a target Info
Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule)
SNS topic

Topic
samson_cap


Amazon EventBridge > Rules > Create rule

Step 1 Define rule detail
Step 2 Build event pattern
Step 3 Select target(s)
Step 4 - optional Configure tags
Step 5 Review and create

Review and create

Step 1: Define rule detail

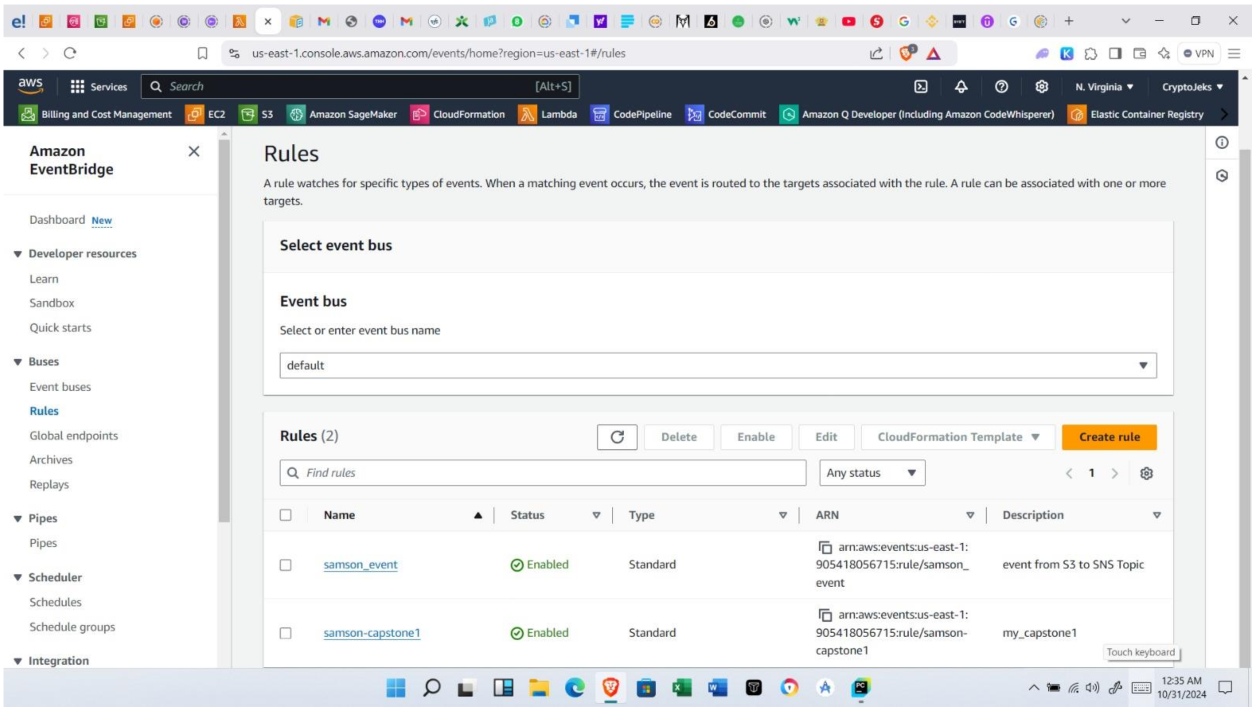
Define rule detail

| Rule name | Status | Event bus |
|---|---|---|
| samson_event | Enabled | default |
| Description | Rule type | |
| event from S3 to SNS Topic | Standard rule | |

Step 2: Build event pattern

Event pattern

```
{
  "source": ["aws.s3"],
  "detail-type": ["Object Created", "Object Deleted", "Object ACL Updated", "Object Restore Initiated"]
}
```


Rules

A rule watches for specific types of events. When a matching event occurs, the event is routed to the targets associated with the rule. A rule can be associated with one or more targets.

Select event bus

Event bus

Select or enter event bus name

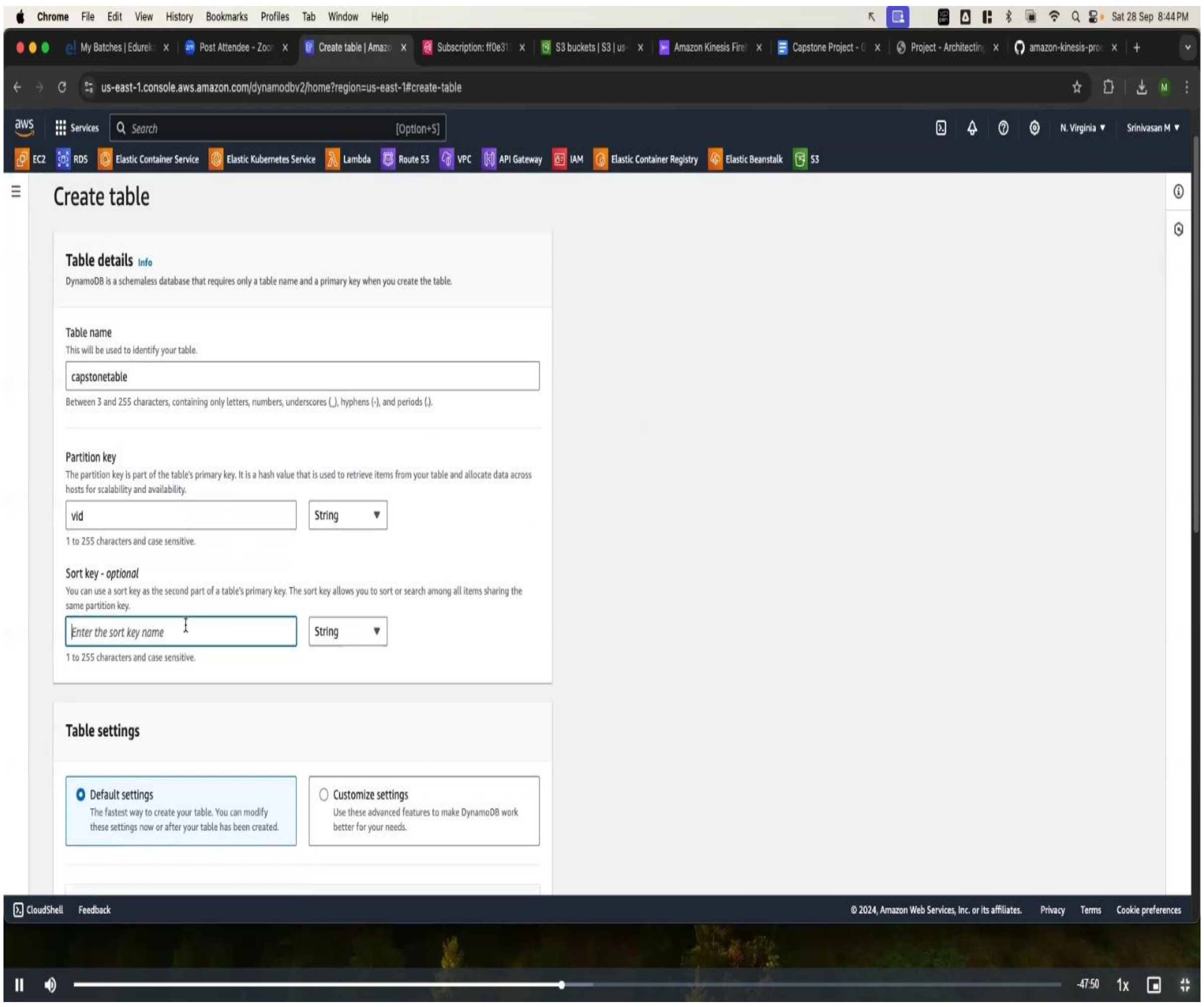
default

Rules (2)

| Name | Status | Type | ARN | Description |
|---|---|---|---|---|
| samson_event | Enabled | Standard | arn:aws:events:us-east-1:905418056715:rule/samson_event | event from S3 to SNS Topic |
| samson-capstone1 | Enabled | Standard | arn:aws:events:us-east-1:905418056715:rule/samson-capstone1 | my_capstone1 |

## DynamoDB Creation

DynamoDB is a fully managed, key-value, and document database that delivers single-digit-millisecond performance at any scale. In this case, my event, stored in my S3 bucket, is pushed on DynamoDB using an SNS Topic via a Lambda function. Below is a screenshot of my DynamoDB creation.


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Sat 28 Sep 8:44 PM
us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#create-table
Services
Search [Option+S]
N. Virginia
Srinivasan M
EC2 RDS Elastic Container Service Elastic Kubernetes Service Lambda Route 53 VPC API Gateway IAM Elastic Container Registry Elastic Beanstalk S3
CloudShell Feedback
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Create table

**Table details** Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

capstonetable

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

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 and allocate data across hosts for scalability and availability.

vid — String

1 to 255 characters and case sensitive.

Sort key - *optional*

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

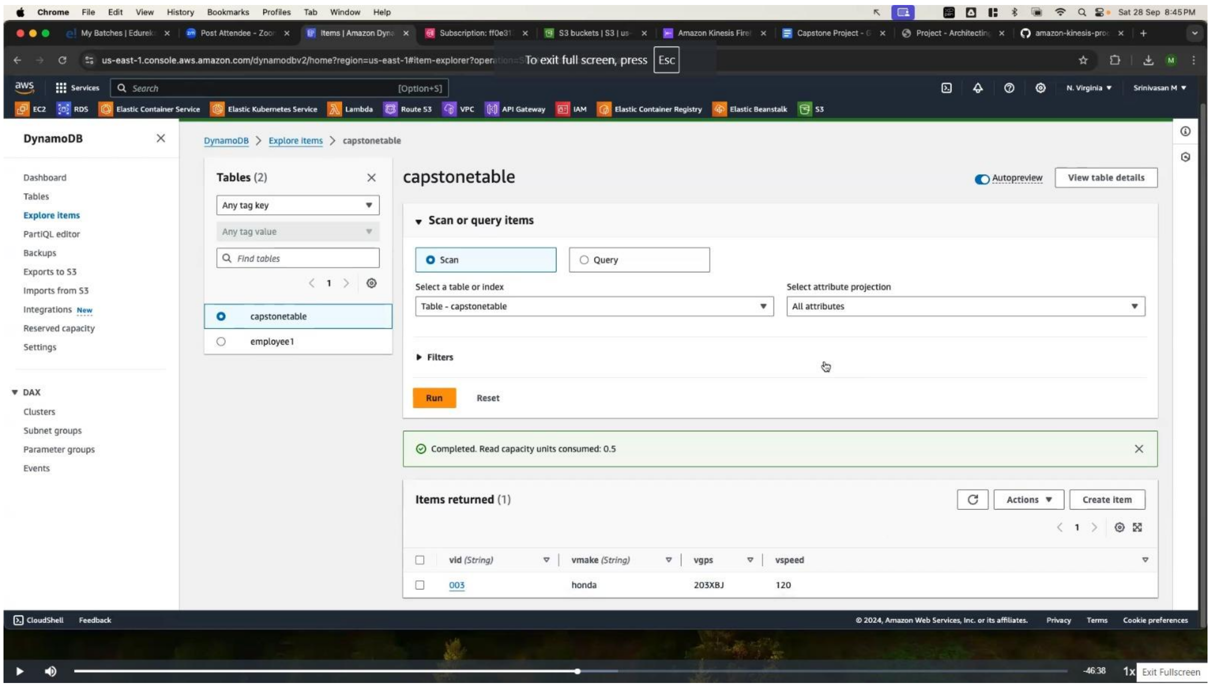
*Enter the sort key name* — String

1 to 255 characters and case sensitive.

**Table 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.



DynamoDB > Explore items > capstonetable

Tables (2)

- capstonetable [✓]
- employee1

## capstonetable

Scan or query items

[✓ Scan] Query

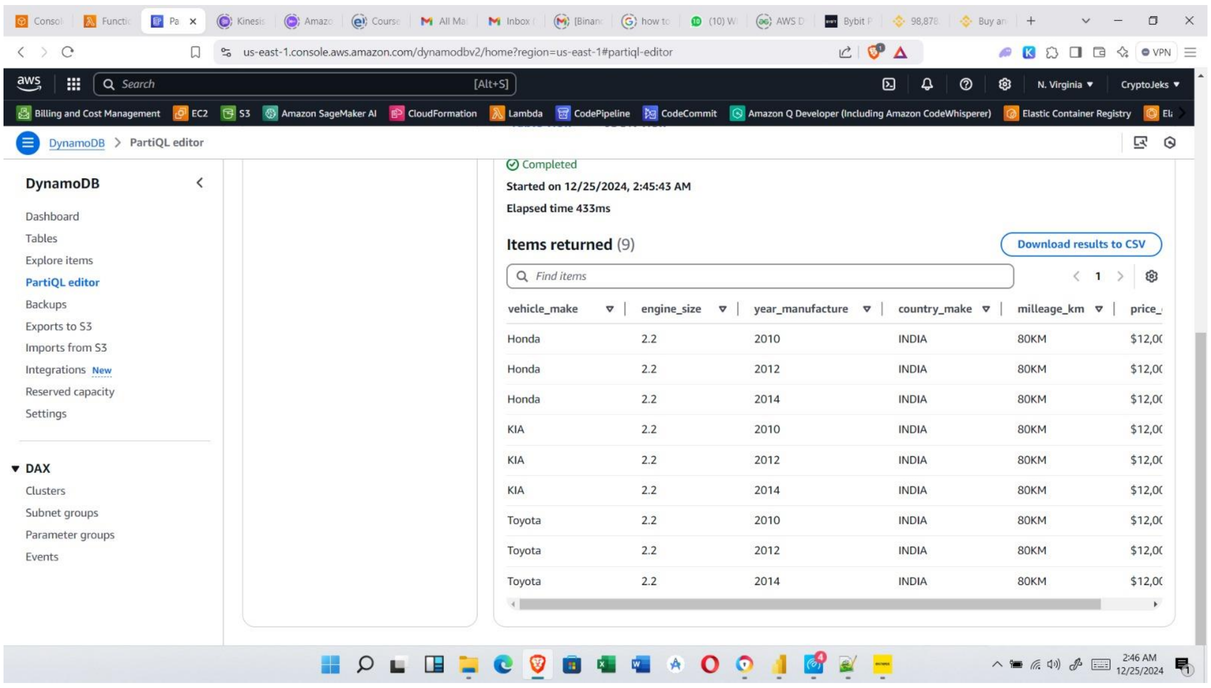
Select a table or index: Table - capstonetable

Select attribute projection: All attributes

Completed. Read capacity units consumed: 0.5

Items returned (1)

| vid (String) | vmake (String) | vgps | vspeed |
|---|---|---|---|
| 003 | honda | 203XBJ | 120 |



DynamoDB > PartiQL editor

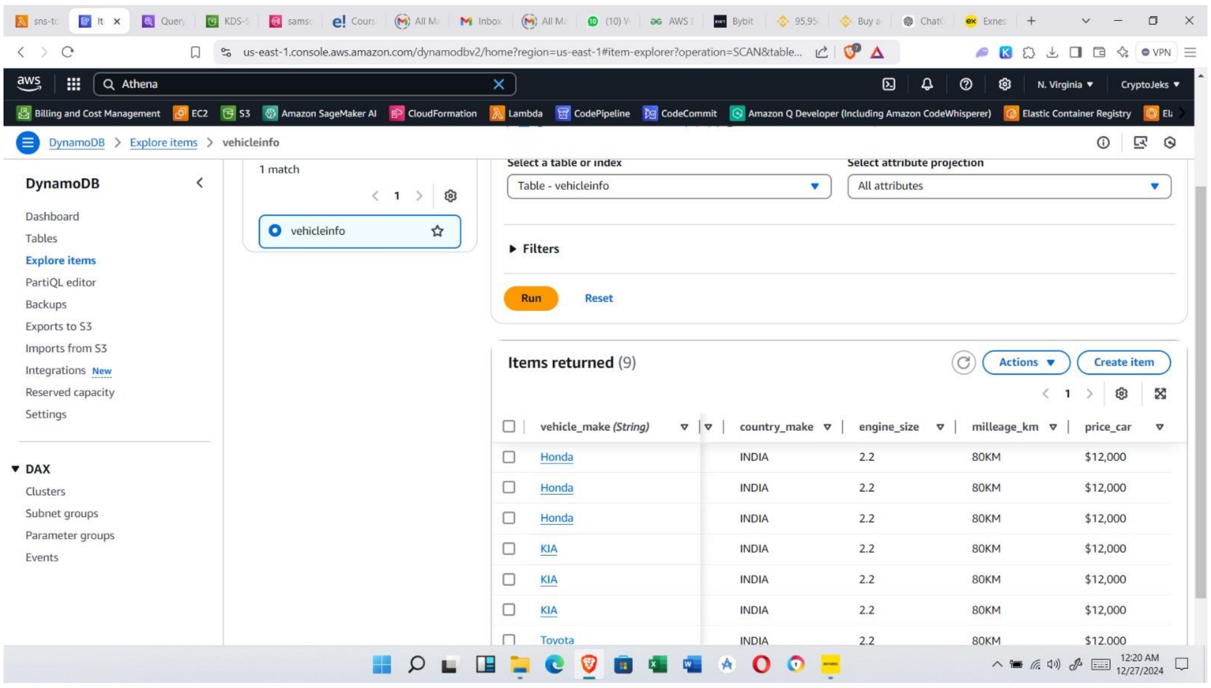
Completed

Started on 12/25/2024, 2:45:43 AM

Elapsed time 433ms

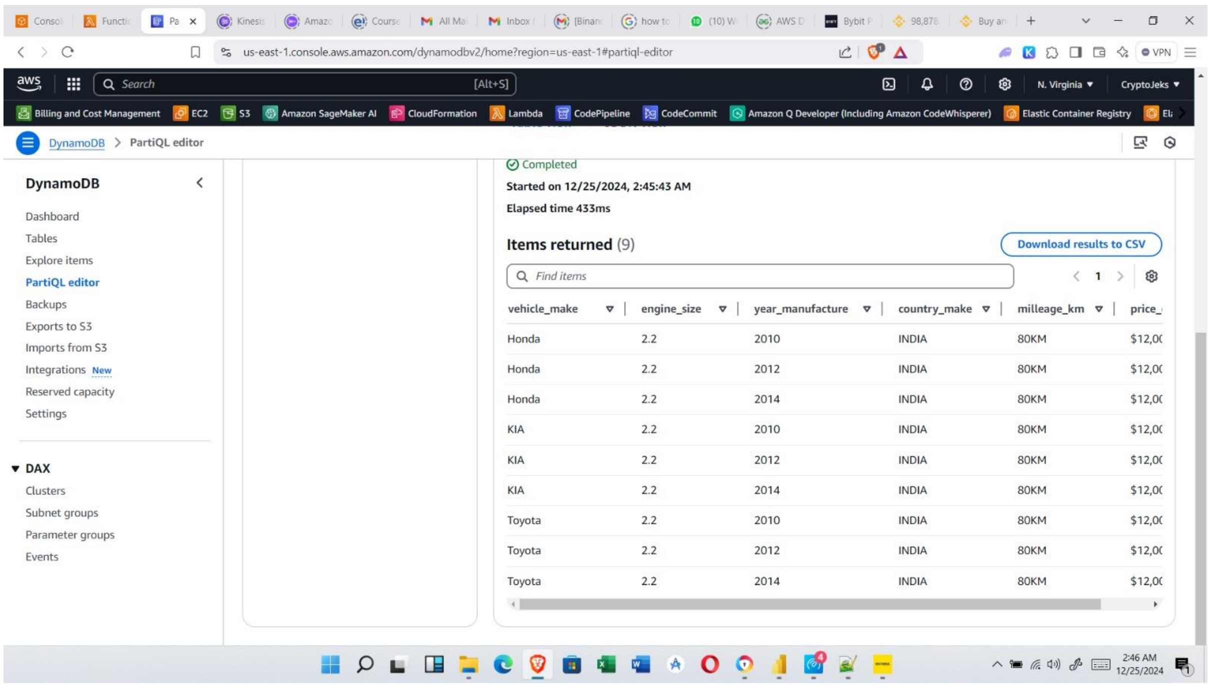
Items returned (9)

| vehicle_make | engine_size | year_manufacture | country_make | mileage_km | price_ |
|---|---|---|---|---|---|
| Honda | 2.2 | 2010 | INDIA | 80KM | $12,0( |
| Honda | 2.2 | 2012 | INDIA | 80KM | $12,0( |
| Honda | 2.2 | 2014 | INDIA | 80KM | $12,0( |
| KIA | 2.2 | 2010 | INDIA | 80KM | $12,0( |
| KIA | 2.2 | 2012 | INDIA | 80KM | $12,0( |
| KIA | 2.2 | 2014 | INDIA | 80KM | $12,0( |
| Toyota | 2.2 | 2010 | INDIA | 80KM | $12,0( |
| Toyota | 2.2 | 2012 | INDIA | 80KM | $12,0( |
| Toyota | 2.2 | 2014 | INDIA | 80KM | $12,0( |



Items returned (9)

| vehicle_make (String) | | country_make | engine_size | milleage_km | price_car |
|---|---|---|---|---|---|
| Honda | | INDIA | 2.2 | 80KM | $12,000 |
| Honda | | INDIA | 2.2 | 80KM | $12,000 |
| Honda | | INDIA | 2.2 | 80KM | $12,000 |
| KIA | | INDIA | 2.2 | 80KM | $12,000 |
| KIA | | INDIA | 2.2 | 80KM | $12,000 |
| KIA | | INDIA | 2.2 | 80KM | $12,000 |
| Toyota | | INDIA | 2.2 | 80KM | $12,000 |



Completed

Started on 12/25/2024, 2:45:43 AM

Elapsed time 433ms

Items returned (9)

| vehicle_make | engine_size | year_manufacture | country_make | milleage_km | price_ |
|---|---|---|---|---|---|
| Honda | 2.2 | 2010 | INDIA | 80KM | $12,0 |
| Honda | 2.2 | 2012 | INDIA | 80KM | $12,0 |
| Honda | 2.2 | 2014 | INDIA | 80KM | $12,0 |
| KIA | 2.2 | 2010 | INDIA | 80KM | $12,0 |
| KIA | 2.2 | 2012 | INDIA | 80KM | $12,0 |
| KIA | 2.2 | 2014 | INDIA | 80KM | $12,0 |
| Toyota | 2.2 | 2010 | INDIA | 80KM | $12,0 |
| Toyota | 2.2 | 2012 | INDIA | 80KM | $12,0 |
| Toyota | 2.2 | 2014 | INDIA | 80KM | $12,0 |

My S3 Event published on my DynamoDB in a Tabular Form

## Amazon Athena

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 and other federated data sources using standard SQL. In this case, I used Amazon Athena to analyze the above data published on my DynamoDB.

## API Gateway Section

Amazon API Gateway helps developers to create and manage APIs to back-end systems running on Amazon EC2, AWS Lambda, or any publicly addressable web service. With Amazon API Gateway, you can generate custom client SDKs for your APIs, to connect your back-end systems to mobile, web, and server applications or services. In this case, I configured Amazon API Gateway to enable external users to have access to the real time data published on DynamoDB via Lambda function. Users can access this data by first, validating their authentication using Amazon Cognito after which, they can connect to the API Gateway either by using Data Lake Console or Data Lake CLI.