Project: Building Ingest-Transform-Load Streaming Pipeline

link for short youtube video:

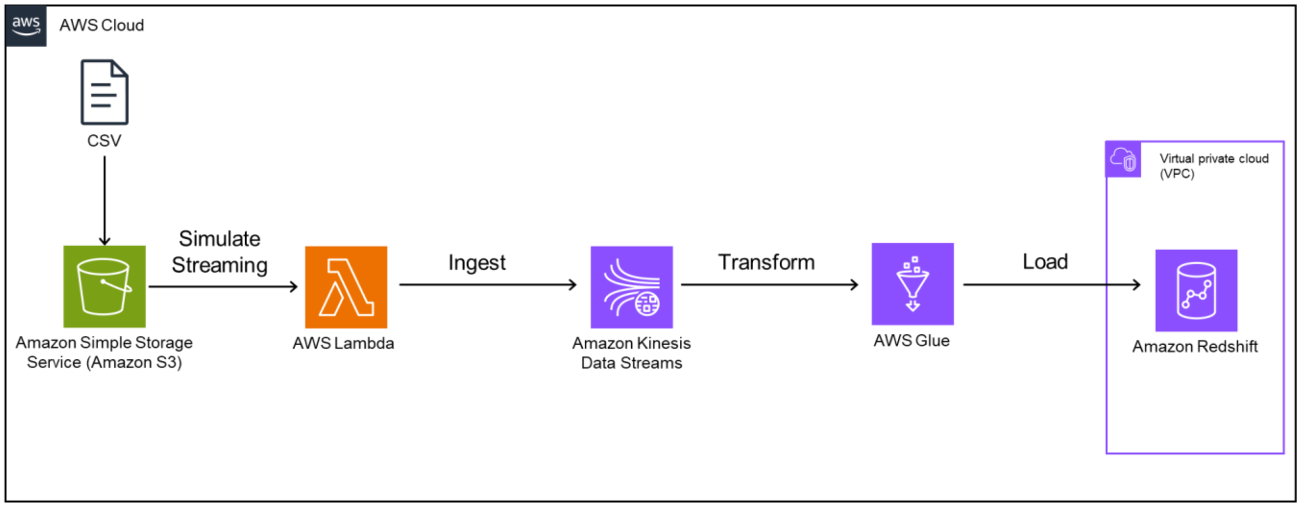
https://youtu.be/B6mv_ffa0Pc

link for long youtube videos:

https://youtu.be/p9z2FUVVbIU

Project Overview

Kinesis Data Stream is a service that offers to process data ingestion that is high in volume and velocity from various data sources and deliver to the desired end point. Kinesis Data Stream also offers fault-tolerant and data distribution across shards based on a partition key, which makes it suitable to be used in ETL process for streaming data. In this demo, we are going to demonstrate the use of Kinesis Data Stream in the ingest-transform-load streaming pipeline of three baseball players three months game statistics, including Shohei Ohtani, Betts Mookie, and Freddie Freeman. The statistics CSV file will be saved in S3 bucket. Lambda is used to stream each record to the Kinesis Data Streaming. After ingestion, the data will be transformed in Glue by selecting only the important statistics. After that, data will be loaded into a warehouse, Amazon redshift.

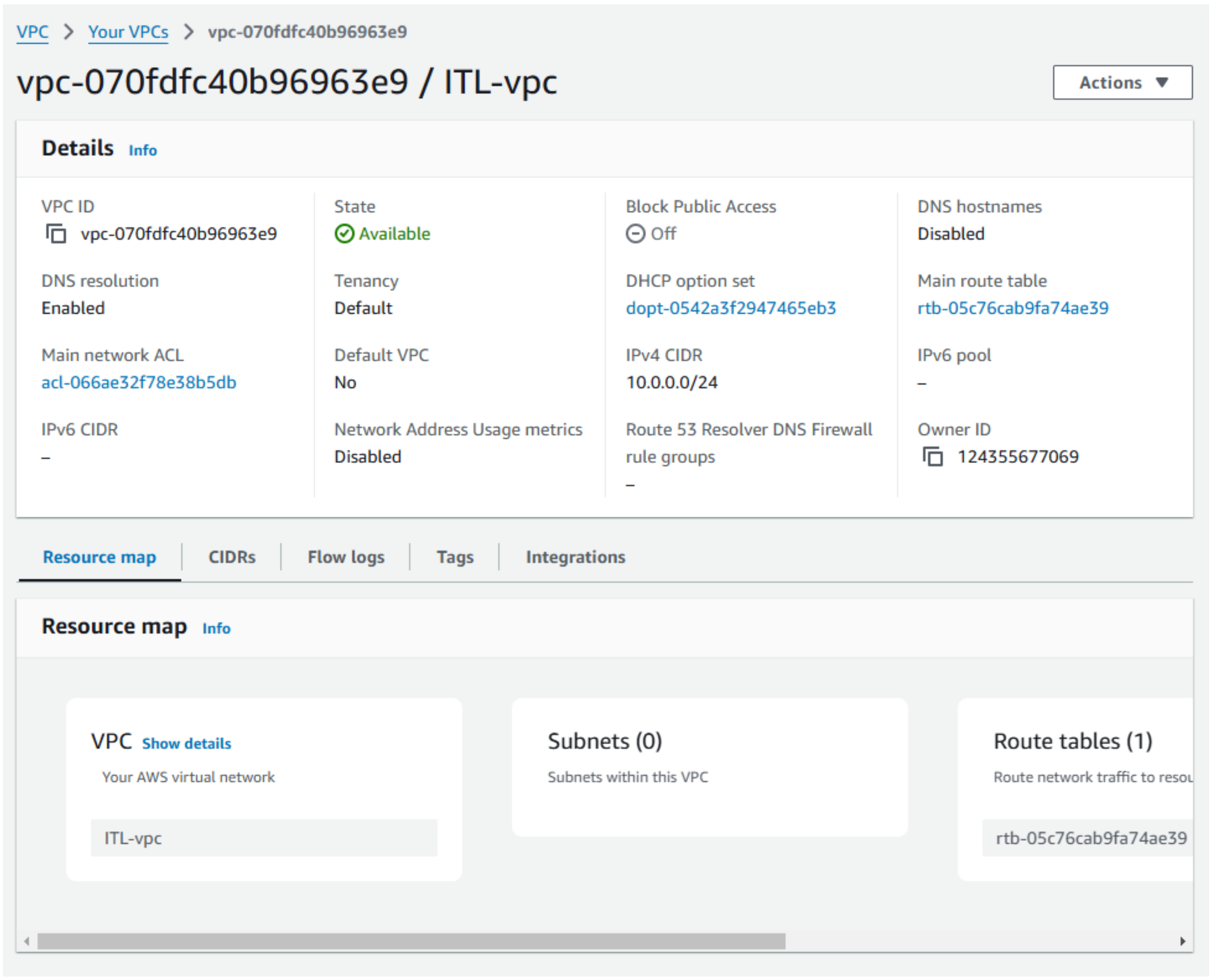

AWS Cloud
CSV
Simulate Streaming
Amazon Simple Storage Service (Amazon S3)
AWS Lambda
Ingest
Amazon Kinesis Data Streams
Transform
AWS Glue
Load
Virtual private cloud (VPC)
Amazon Redshift

Create a Virtual Private Cloud and Private Subnets

The VPC and subnets are created to only host Amazon Redshift

- Go to VPC service in AWS
- Click on Create VPC
- Name it ITL-vpc
- Choose IPv4 CIDR manual input
- For IPv4 CIDR, use 10.0.0.0/24 for amount of 4,096 IPs
- Select No IPv6 CIDR block
- Click create VPC

Your VPC list should look as the following:


VPC > Your VPCs > vpc-070fdfc40b96963e9

vpc-070fdfc40b96963e9 / ITL-vpc

Actions

Details Info

VPC ID	State	Block Public Access	DNS hostnames
vpc-070fdfc40b96963e9	Available	Off	Disabled
DNS resolution	Tenancy	DHCP option set	Main route table
Enabled	Default	dopt-0542a3f2947465eb3	rtb-05c76cab9fa74ae39
Main network ACL	Default VPC	IPv4 CIDR	IPv6 pool
acl-066ae32f78e38b5db	No	10.0.0.0/24	–
IPv6 CIDR	Network Address Usage metrics	Route 53 Resolver DNS Firewall rule groups	Owner ID
–	Disabled	–	124355677069

Resource map Info

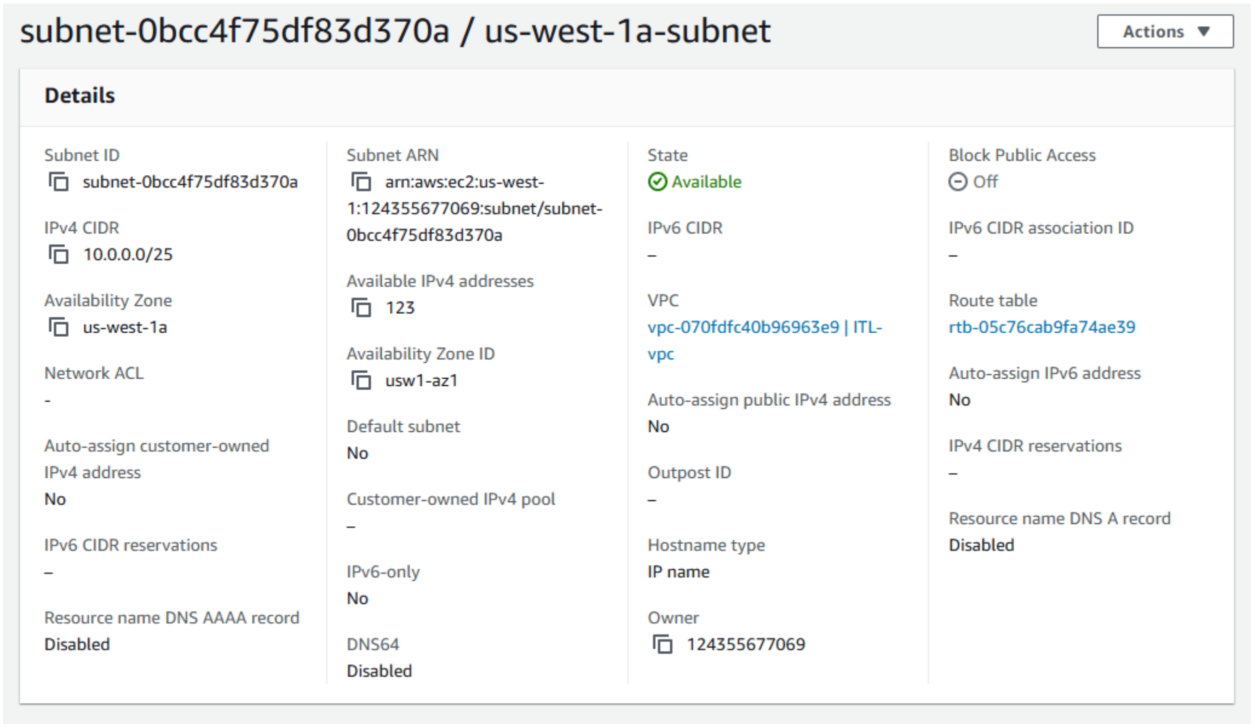
VPC Show details
Your AWS virtual network
ITL-vpc

Subnets (0)
Subnets within this VPC

Route tables (1)
Route network traffic to resou
rtb-05c76cab9fa74ae39

- Go to subnets
- Click create subnet
- Name subnet us-west-1a-subnet
- Choose the Availability Zone in your area. I choose us-west-1a
- Choose 10.0.0.0/24 for IPv4 VPC CIDR block
- Choose 10.0.0.0/25 for IPv4 subnet CIDR block with 128 IP addresses
- Create subnet

Your subnet should look as the following:



subnet-0bcc4f75df83d370a / us-west-1a-subnet

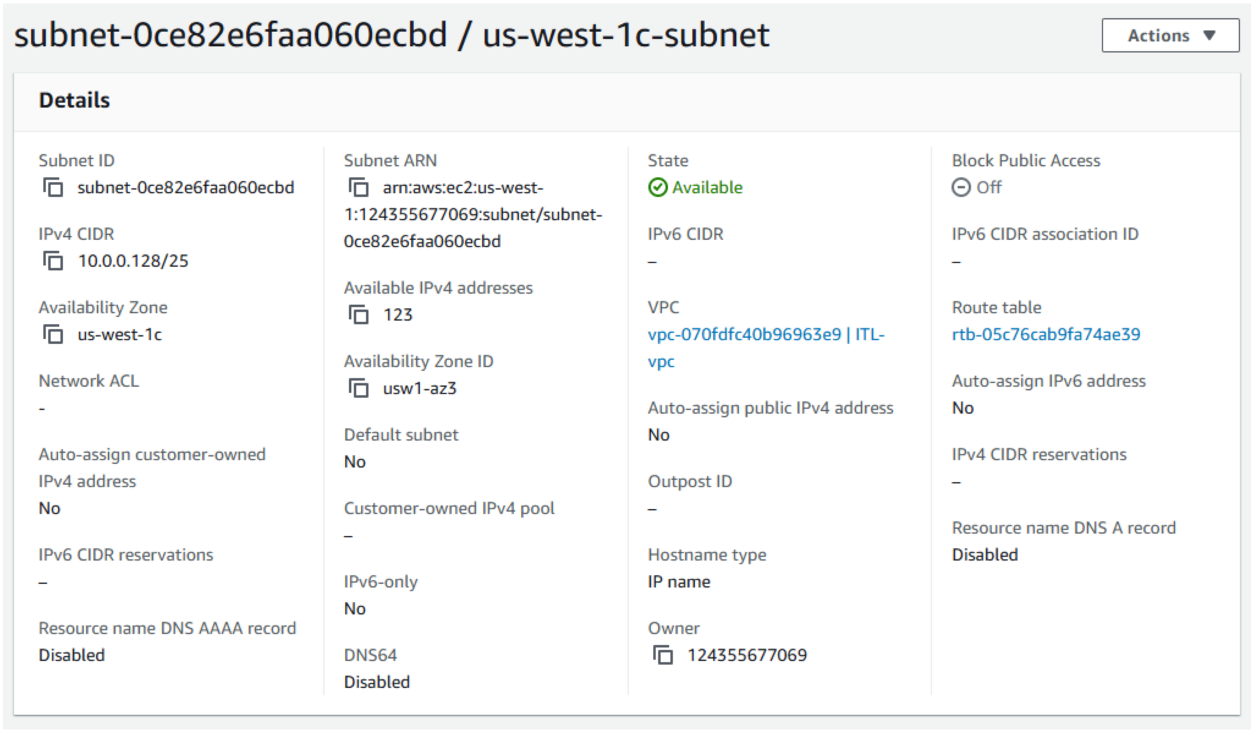
Details

Subnet ID	Subnet ARN	State	Block Public Access
subnet-0bcc4f75df83d370a	arn:aws:ec2:us-west-1:124355677069:subnet/subnet-0bcc4f75df83d370a	Available	Off

IPv4 CIDR: 10.0.0.0/25	Available IPv4 addresses: 123	IPv6 CIDR: –	IPv6 CIDR association ID: –
Availability Zone: us-west-1a	Availability Zone ID: usw1-az1	VPC: vpc-070fdfc40b96963e9 | ITL-vpc	Route table: rtb-05c76cab9fa74ae39
Network ACL: -	Default subnet: No	Auto-assign public IPv4 address: No	Auto-assign IPv6 address: No
Auto-assign customer-owned IPv4 address: No	Customer-owned IPv4 pool: –	Outpost ID: –	IPv4 CIDR reservations: –
IPv6 CIDR reservations: –	IPv6-only: No	Hostname type: IP name	Resource name DNS A record: Disabled
Resource name DNS AAAA record: Disabled	DNS64: Disabled	Owner: 124355677069	

- Create another subnet name us-west-1c-subnet
- Choose the Availability Zone in your area. I choose us-west-1c
- Choose 10.0.0.0/24 for IPv4 VPC CIDR block
- Choose 10.0.0.128/25 for IPv4 subnet CIDR block with 128 IP address

Your subnet should look as the following:



subnet-0ce82e6faa060ecbd / us-west-1c-subnet

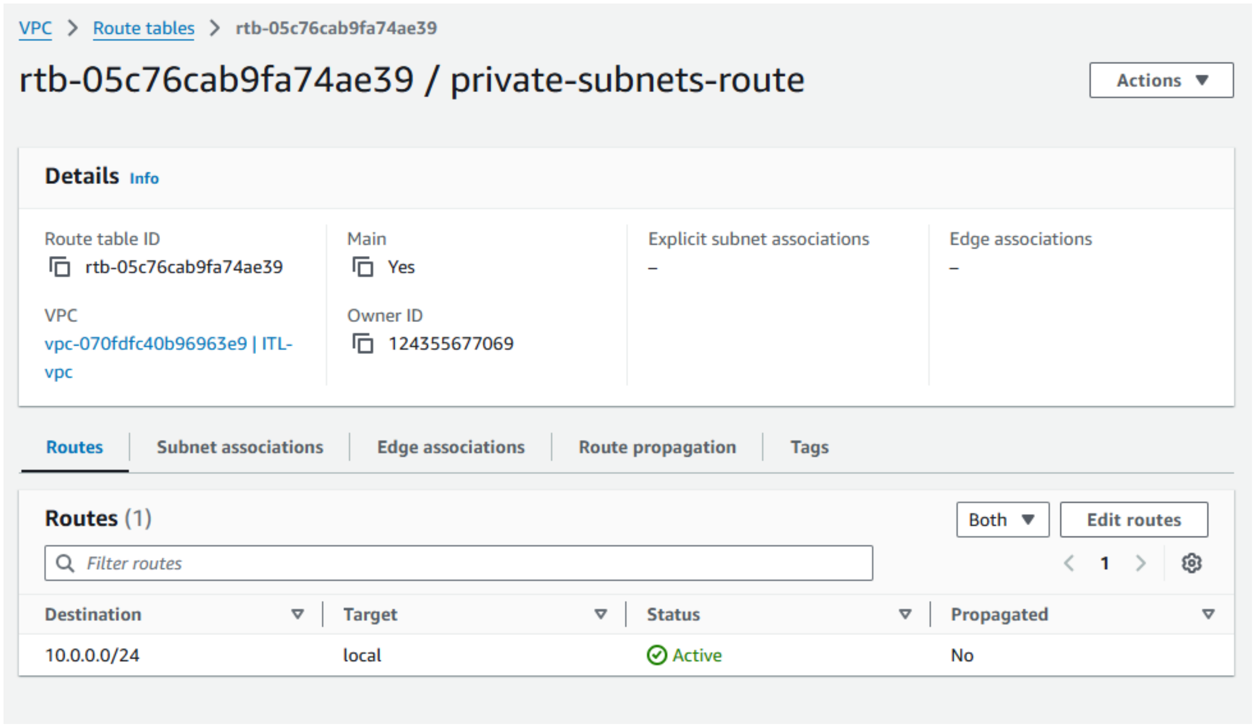
Details

Subnet ID	Subnet ARN	State	Block Public Access
subnet-0ce82e6faa060ecbd	arn:aws:ec2:us-west-1:124355677069:subnet/subnet-0ce82e6faa060ecbd	Available	Off

IPv4 CIDR: 10.0.0.128/25	Available IPv4 addresses: 123	IPv6 CIDR: –	IPv6 CIDR association ID: –
Availability Zone: us-west-1c	Availability Zone ID: usw1-az3	VPC: vpc-070fdfc40b96963e9 | ITL-vpc	Route table: rtb-05c76cab9fa74ae39
Network ACL: -	Default subnet: No	Auto-assign public IPv4 address: No	Auto-assign IPv6 address: No
Auto-assign customer-owned IPv4 address: No	Customer-owned IPv4 pool: –	Outpost ID: –	IPv4 CIDR reservations: –
IPv6 CIDR reservations: –	IPv6-only: No	Hostname type: IP name	Resource name DNS A record: Disabled
Resource name DNS AAAA record: Disabled	DNS64: Disabled	Owner: 124355677069	

- Check in route tables, there should be a default route tables created

- Name it private-subnets-route
- 10.0.0.0/24 allows routing inside the VPC



rtb-05c76cab9fa74ae39 / private-subnets-route

Details Info

Route table ID	Main	Explicit subnet associations	Edge associations
rtb-05c76cab9fa74ae39	Yes	–	–
VPC	Owner ID		
vpc-070fdfc40b96963e9 | ITL-vpc	124355677069		

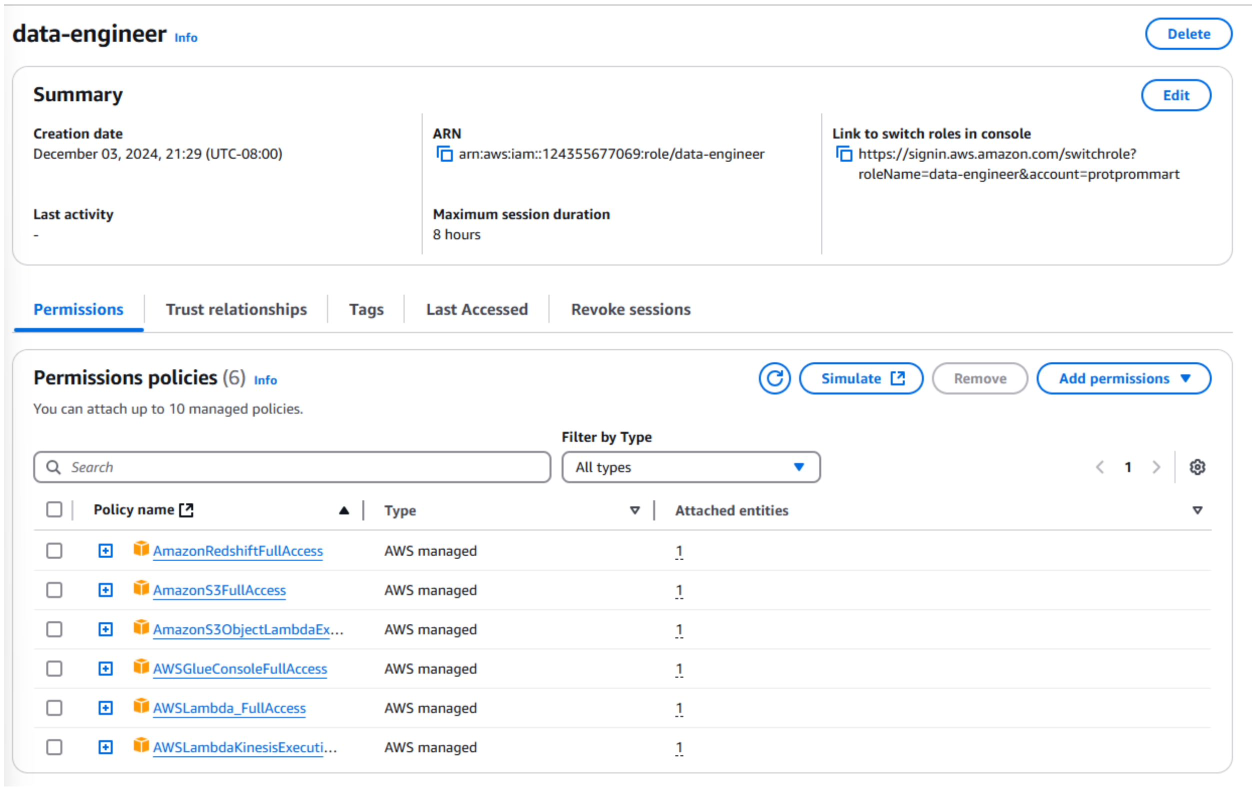
Routes (1)

Destination	Target	Status	Propagated
10.0.0.0/24	local	Active	No

Create IAM role for the demo

- Go to IAM and choose roles
- Click create role
- Choose AWS account for select trusted entity and choose this account
- Attach the following policies:
  - AmazonS3ObjectLambdaExecutionRolePolicy for S3 to interact with Lambda
  - AWS Lambda_fullAccess
  - AWSS3FullAccess
  - AWSLambdaKinesisExecutionRole
  - AmazonKinesisFullAccess
  - AmazonRedshiftFullAccess
  - AWSGlueConsoleFullAccess
- Name Role Name data-engineer
- Create role

You should be able to see your role as this:



data-engineer Info

Summary

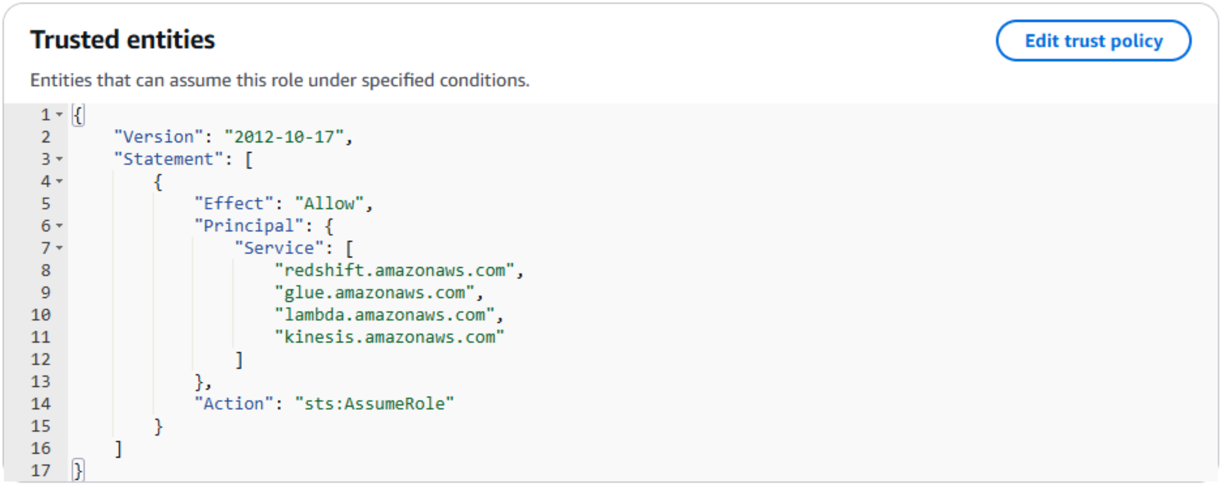
Creation date	ARN	Link to switch roles in console
December 03, 2024, 21:29 (UTC-08:00)	arn:aws:iam::124355677069:role/data-engineer	https://signin.aws.amazon.com/switchrole?roleName=data-engineer&account=protprommart
Last activity	**Maximum session duration**	
-	8 hours	

Permissions policies (6) Info

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonRedshiftFullAccess	AWS managed	1
AmazonS3FullAccess	AWS managed	1
AmazonS3ObjectLambdaEx...	AWS managed	1
AWSGlueConsoleFullAccess	AWS managed	1
AWSLambda_FullAccess	AWS managed	1
AWSLambdaKinesisExecuti...	AWS managed	1

- Once done, go to the trust relationship tab of the data engineer role we created.
- We are going to add services that we want to use this role with including redshift, lambda, glue, kinesis.
- In the json documents, use the following json to establish service trust



Trusted entities

Entities that can assume this role under specified conditions.

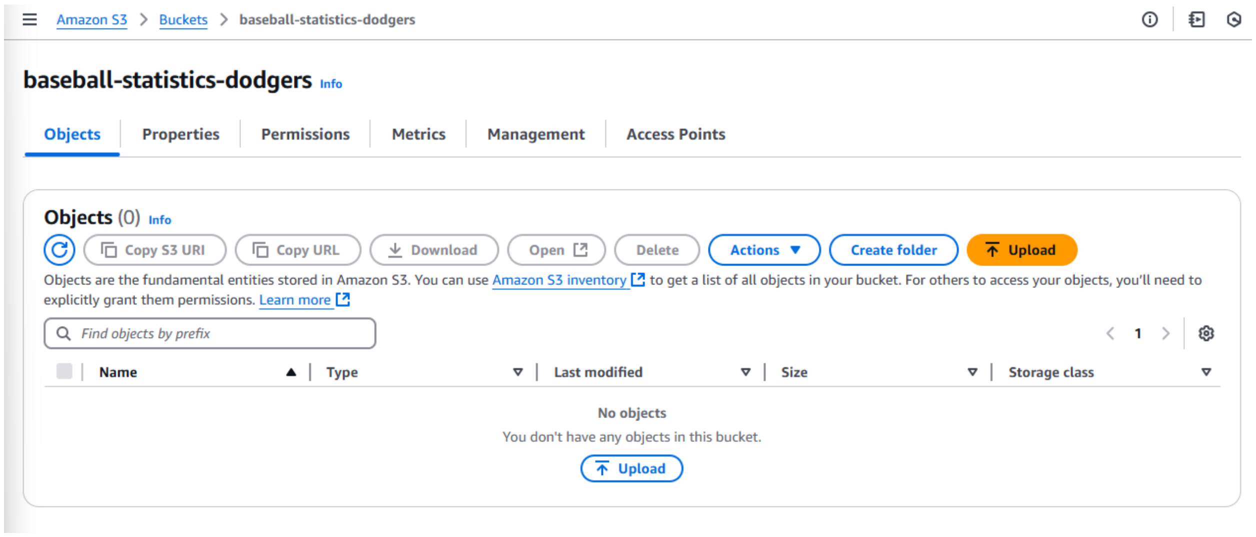
```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "redshift.amazonaws.com",
                    "glue.amazonaws.com",
                    "lambda.amazonaws.com",
                    "kinesis.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Create a S3 bucket

- Go to Amazon S3 bucket in AWS
- Click on Create bucket
- Name the bucket baseball-statistics-dodgers

- Choose ACLs disabled for object ownership
- Select block all public access
- Choose disable bucket Versioning
- Create bucket

You should have a blank bucket like this:



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

No objects
You don't have any objects in this bucket.

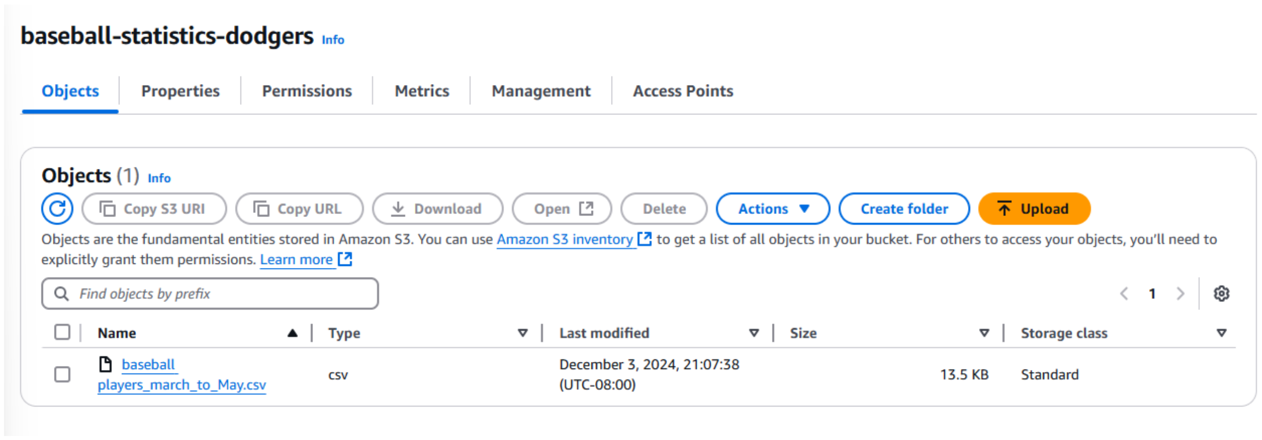
<u>Input Baseball Statistics into S3 bucket</u>

I extracted game log statistics from March to May 2024 of LA Dodgers players Shohei Ohtani, Betts Mookie, and Freddie Freeman from LA Dodgers websites:
https://www.mlb.com/player/freddie-freeman-518692?season=2024&team=119&stats=gamelogs-r-hitting-mlb&year=2024
https://www.mlb.com/player/shohei-ohtani-660271?stats=gamelogs-r-hitting-mlb&year=2024
https://www.mlb.com/player/mookie-betts-605141?season=2024&team=119&stats=gamelogs-r-hitting-mlb&year=2024

The csv file of the game log statistics from March to May 2024 can be found in my github repository link: https://github.com/Protprommart/LA-Dodgers-Cloud-ETL/blob/main/baseball_players_march_to_may.csv

- Download csv file from the github link
- Upload this file into the baseball-statistics-dodgers S3 bucket

You should be able to see the file in the bucket like this:

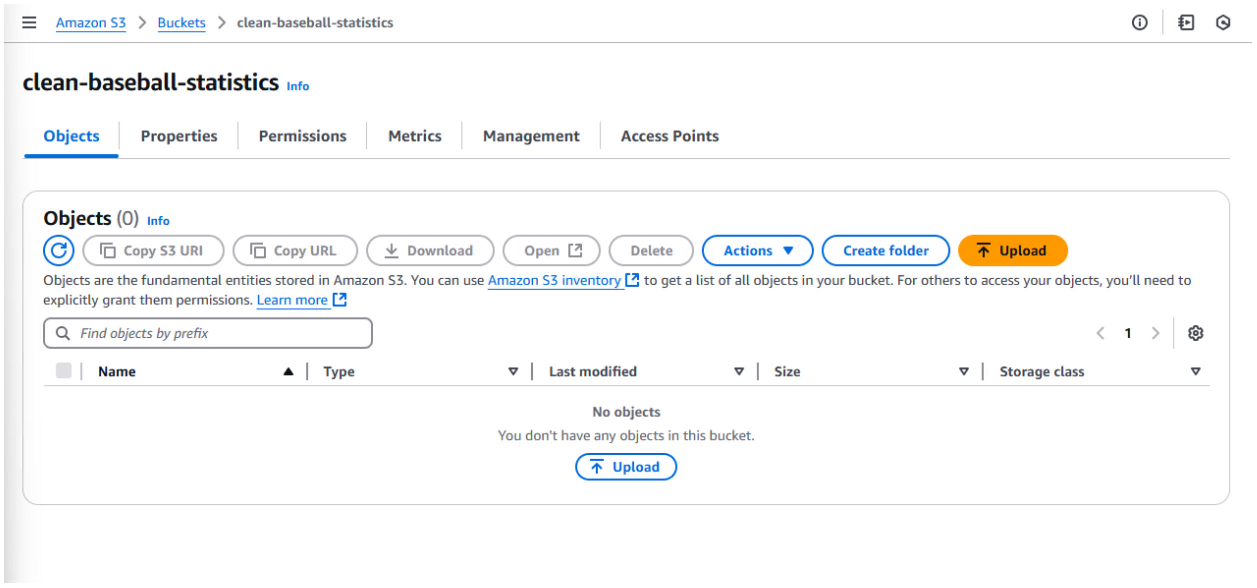

baseball-statistics-dodgers Info

Objects (1) 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
baseball players_march_to_May.csv	csv	December 3, 2024, 21:07:38 (UTC-08:00)	13.5 KB	Standard

- Create another S3 bucket called clean-baseball-statistics with the same instruction from before. This is going to be a bucket where we put in our transformed dataset after running the data with visual ETL. You should have the bucket as follow:


Amazon S3 > Buckets > clean-baseball-statistics

clean-baseball-statistics Info

Objects (0) 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

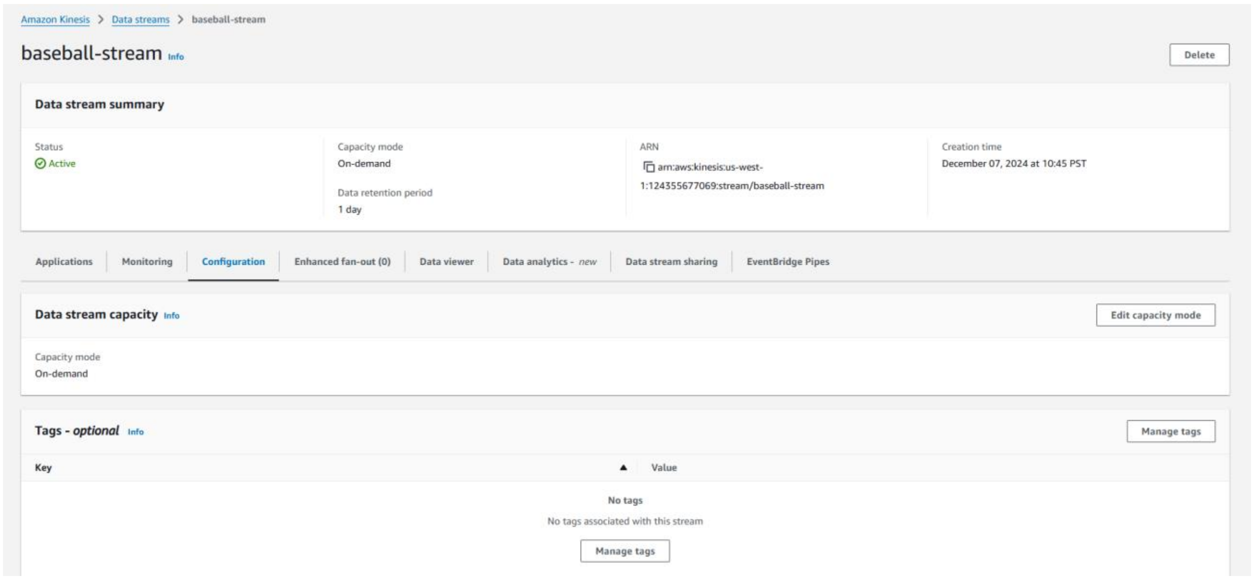
No objects

You don't have any objects in this bucket.

Set Up Kinesis Data Stream and Streaming through Lambda

Create a Kinesis Data Stream

- Go to the Amazon Kinesis Console
- Go to Data Streams.
- Create a new stream name baseball-stream
- Leave other setting as default
- Your Kinesis Data Stream should look as following:


Amazon Kinesis > Data streams > baseball-stream

baseball-stream Info

Data stream summary

Status: Active
Capacity mode: On-demand
Data retention period: 1 day
ARN: arn:aws:kinesis:us-west-1:124355677069:stream/baseball-stream
Creation time: December 07, 2024 at 10:45 PST

Data stream capacity Info

Capacity mode: On-demand

Tags - *optional* Info

Key	Value

No tags
No tags associated with this stream

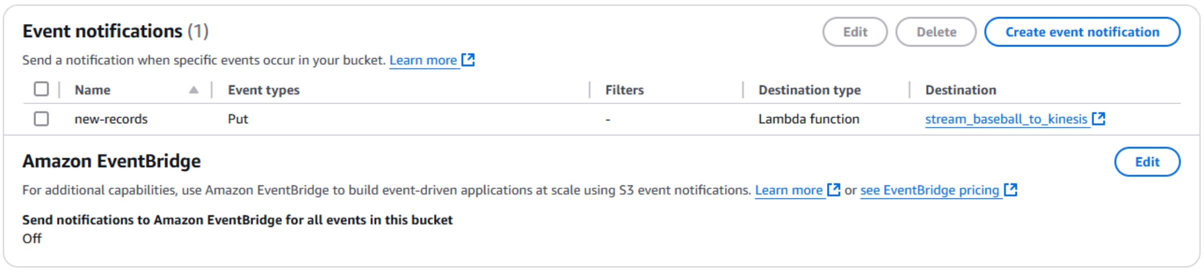
Create a Lambda Function

- Navigate to the Lambda Console
- Create Function.
- Select "Author from scratch".
    - Function Name: stream_baseball_to_kinesis.
    - Runtime: Python 3.13.
    - Execution Role: Use the IAM role data engineer
    - Create function
    - Write the Lambda Function Code:
      Below is the Python example for the Lambda function:
    - Click Deploy button after you finish inputting code

```
import boto3
import csv
import json
import os

# Initialize the Kinesis client
kinesis_client = boto3.client('kinesis')

def lambda_handler(event, context):
    try:
        # Get S3 bucket and object key from the event
        bucket_name = event['Records'][0]['s3']['bucket']['name']
        object_key = event['Records'][0]['s3']['object']['key']
        print(f"Bucket: {bucket_name}, Key: {object_key}")

        # Download the file from S3
        s3_client = boto3.client('s3')
        response = s3_client.get_object(Bucket=bucket_name,
Key=object_key)
        data = response['Body'].read().decode('utf-8-sig').splitlines()

        # Parse the CSV file
        reader = csv.DictReader(data)

        # Stream records to Kinesis
        for row in reader:
            #skip row for no player and print it out for audit
            if not row['Player']:
                print(f"Skipping empty row: {row}")
                continue

            # Convert each row to JSON
            record = json.dumps(row)
            print(f"Sending record: {record}")

            # Send record to Kinesis Data Stream
            partition_key = row['Player']
            kinesis_client.put_record(
                StreamName='baseball-stream',
                Data=record,
                PartitionKey=partition_key
            )
        #verify stream complete
        print("Streaming to Kinesis completed.")

    #stream fail and reason for failing
    except Exception as e:
        print(f"Streaming to Kinesis failed: {str(e)}")
```

Configure S3 Event Notifications

- Navigate to the S3 bucket in the AWS Management Console.
- Go to the Properties tab and Event Notifications.
- Create a notification as following:
    - Event name: new-records
    - Event Type: PUT (for new file uploads).
    - Destination: Lambda Function (created in the next step).
    - Lambda function: stream_baseball_to_kinesis.
- Your Kinesis S3 Event Notifications should look as following:



Event notifications (1)

Send a notification when specific events occur in your bucket. Learn more

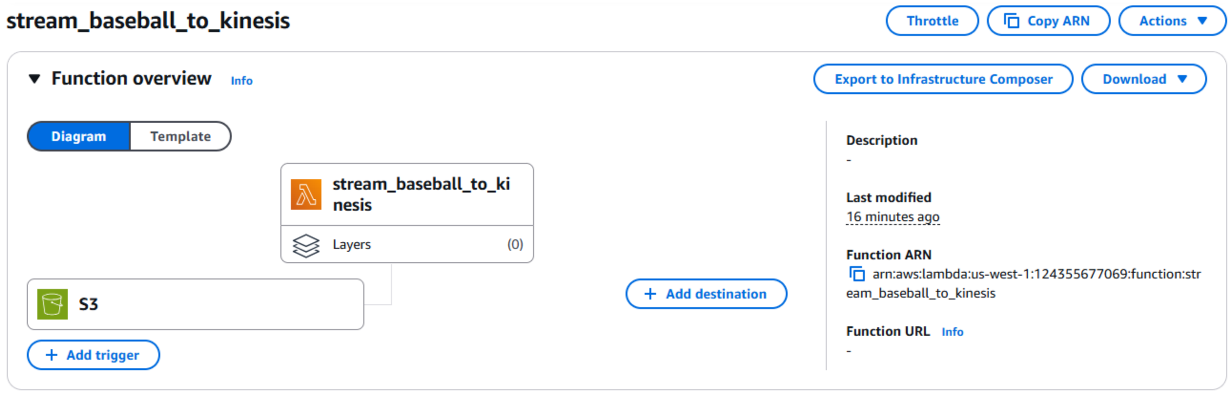
Name	Event types	Filters	Destination type	Destination
new-records	Put	-	Lambda function	stream_baseball_to_kinesis

Amazon EventBridge

For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. Learn more or see EventBridge pricing

Send notifications to Amazon EventBridge for all events in this bucket
Off

- Your Lambda connected to S3 Notification should look as following:



stream_baseball_to_kinesis

Function overview

Description
-

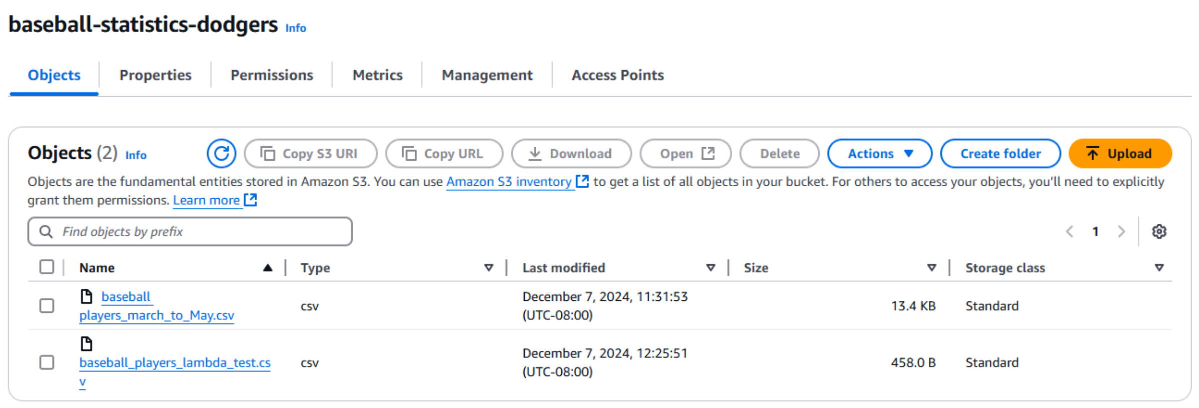
Last modified
16 minutes ago

Function ARN
arn:aws:lambda:us-west-1:124355677069:function:stream_baseball_to_kinesis

Function URL
-

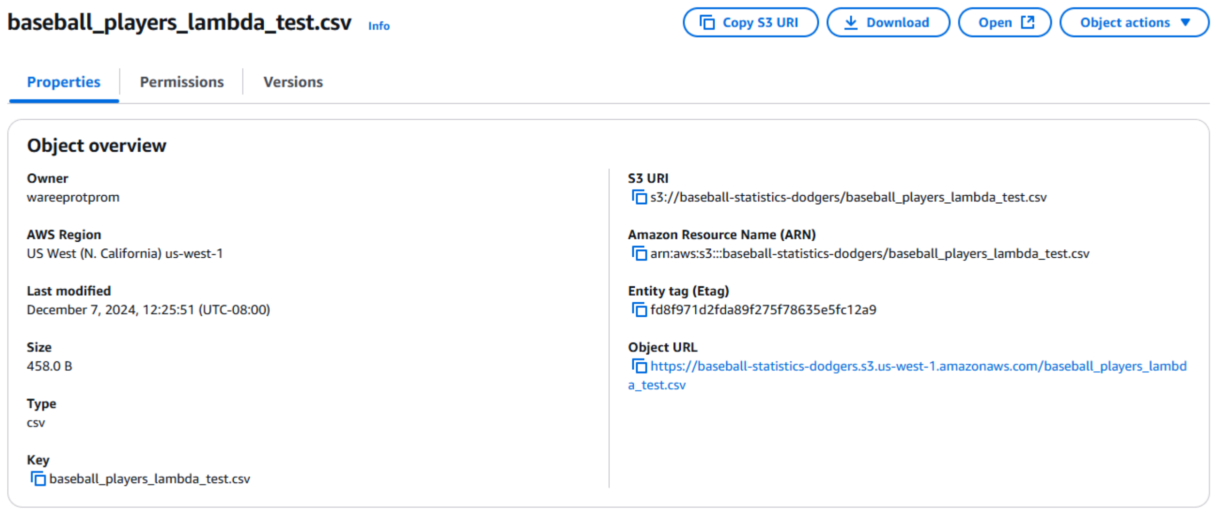
Create a Lambda Test Function

The csv file of the game log statistics for lambda testing can be found in my github repository link:
https://github.com/Protprommart/LA-Dodgers-Cloud-ETL/blob/main/baseball_players_lambda_test.csv
Upload this csv file into baseball-statistics-dodgers S3 bucket as following:



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
baseball players_march_to_May.csv	csv	December 7, 2024, 11:31:53 (UTC-08:00)	13.4 KB	Standard
baseball_players_lambda_test.csv	csv	December 7, 2024, 12:25:51 (UTC-08:00)	458.0 B	Standard



Owner
wareeprotprom

AWS Region
US West (N. California) us-west-1

Last modified
December 7, 2024, 12:25:51 (UTC-08:00)

Size
458.0 B

Type
csv

Key
baseball_players_lambda_test.csv

S3 URI
s3://baseball-statistics-dodgers/baseball_players_lambda_test.csv

Amazon Resource Name (ARN)
arn:aws:s3:::baseball-statistics-dodgers/baseball_players_lambda_test.csv

Entity tag (Etag)
fd8f971d2fda89f275f78635e5fc12a9

Object URL
https://baseball-statistics-dodgers.s3.us-west-1.amazonaws.com/baseball_players_lambda_test.csv

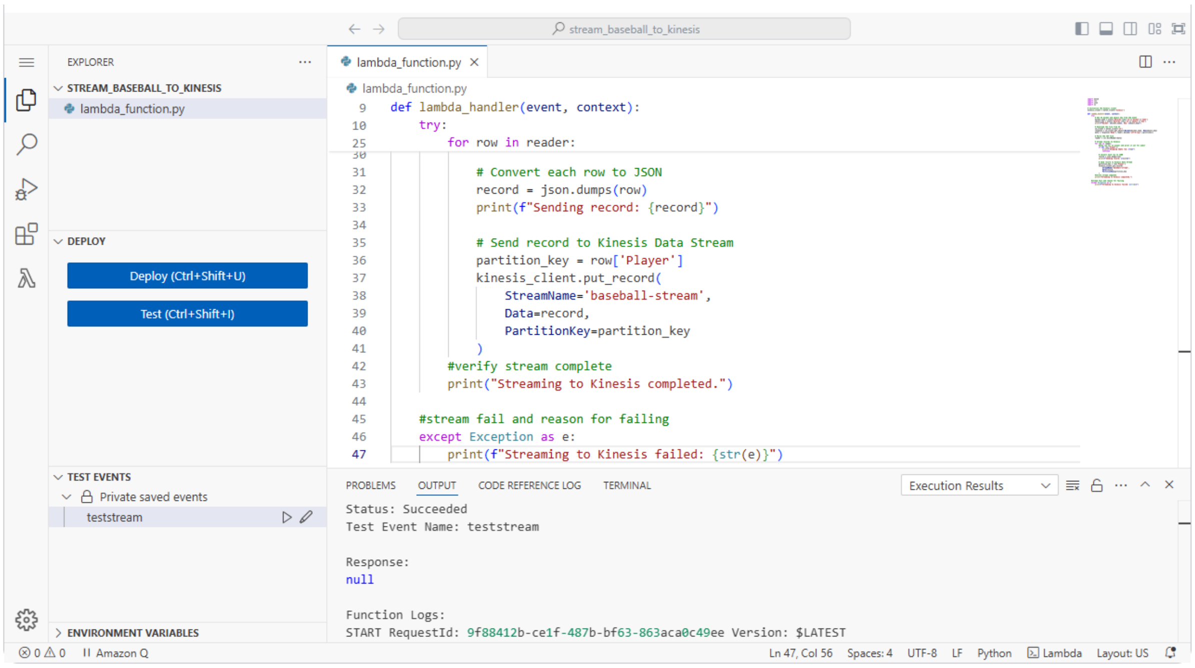
- Click on test button in stream_baseball_to_kinesis lambda function
- Name the test as teststream
- Use the S3 PUT event template.
- Update the event JSON to include:
  - Correct bucket name: baseball-statistics-dodgers.
  - File name (key): baseball_players_lambda_test.csv
  - Put in your awsRegion you are operating from
  - Put in the correct Etag of the object by getting it from S3 bucket object.
- Your test event JSON should look as following:

```
{
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "us-west-1",
      "eventTime": "2023-03-10T15:21:30.000Z",
      "eventName": "ObjectCreated:Put",
      "s3": {
        "s3SchemaVersion": "1.0",
        "bucket": {
          "name": "baseball-statistics-dodgers",
          "arn": "arn:aws:s3:::baseball-statistics-dodgers"
        },
        "object": {
          "key": "baseball_players_lambda_test.csv",
          "size": 801.0,
          "eTag": "6ced031204f54ad963f10e25556936d0",
          "sequencer": "00123456789abcdef"
        }
      }
    }
  ]
}
```

- Run the test code. Your output should be looking as following:

```
Status: Succeeded
Test Event Name: teststream

Response:
null

Function Logs:
START RequestId: 9f88412b-ce1f-487b-bf63-863aca0c49ee Version: $LATEST
Bucket: baseball-statistics-dodgers, Key:
baseball_players_lambda_test.csv
Sending record: {"Player": "Shohei Ohtani", "Team": "20-Mar", "OPP": "@
SD", "At Bats": "5", "Runs": "0", "Hits": "2", "Total Bases": "2", "2B":
"0", "3B": "0", "Home Runs": "0", "RunsBattedIn": "1", "Bases on Balls":
"0", "Intentional Walks": "0", "StrikeOut": "0", "StolenBases": "1",
"CaughtStealing": "0", "AVG": "0.4", "On-Base Percentage": "0.4",
"Slugging Percentage": "0.4", "HitByPitch": "0", "sacrifice bunt": "0",
"Sacrifice Flies": "0"}
Sending record: {"Player": "Shohei Ohtani", "Team": "21-Mar", "OPP": "vs
SD", "At Bats": "5", "Runs": "1", "Hits": "1", "Total Bases": "1", "2B":
```

```
"0", "3B": "0", "Home Runs": "0", "RunsBattedIn": "1", "Bases on Balls":
"0", "Intentional Walks": "0", "StrikeOut": "0", "StolenBases": "0",
"CaughtStealing": "0", "AVG": "0.3", "On-Base Percentage": "0.273",
"Slugging Percentage": "0.3", "HitByPitch": "0", "sacrifice bunt": "0",
"Sacrifice Flies": "1"}
Sending record: {"Player": "Shohei Ohtani", "Team": "28-Mar", "OPP": "vs
STL", "At Bats": "3", "Runs": "1", "Hits": "2", "Total Bases": "3",
"2B": "1", "3B": "0", "Home Runs": "0", "RunsBattedIn": "0", "Bases on
Balls": "1", "Intentional Walks": "0", "StrikeOut": "1", "StolenBases":
"0", "CaughtStealing": "0", "AVG": "0.385", "On-Base Percentage": "0.4",
"Slugging Percentage": "0.462", "HitByPitch": "0", "sacrifice bunt":
"0", "Sacrifice Flies": "0"}
Streaming to Kinesis completed.
END RequestId: 9f88412b-ce1f-487b-bf63-863aca0c49ee
REPORT RequestId: 9f88412b-ce1f-487b-bf63-863aca0c49ee  Duration:
2125.70 ms    Billed Duration: 2126 ms    Memory Size: 128 MB Max Memory
Used: 85 MB  Init Duration: 538.16 ms

Request ID: 9f88412b-ce1f-487b-bf63-863aca0c49ee
```



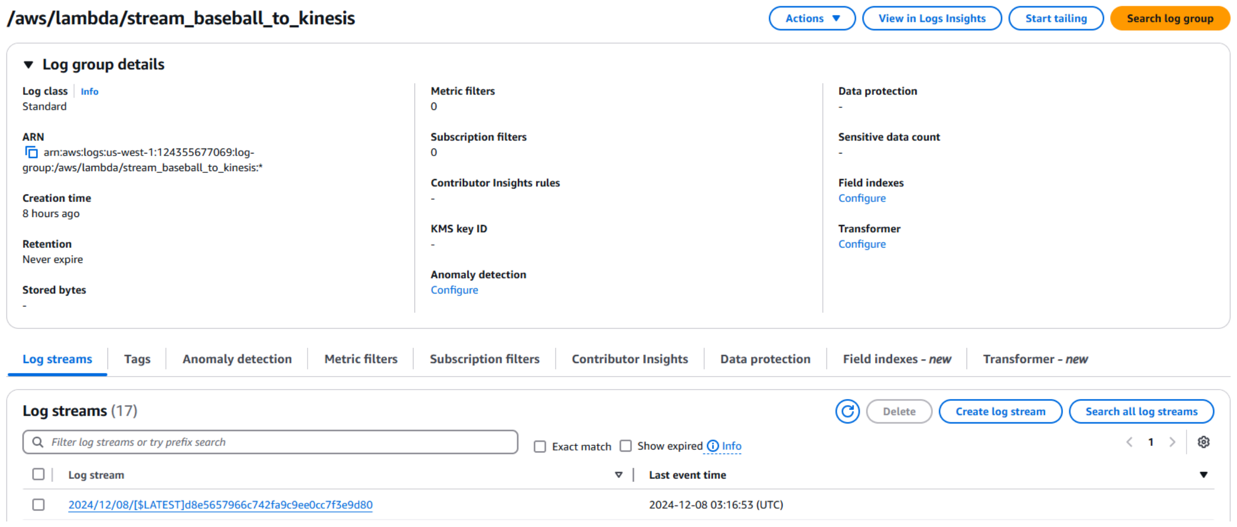
```python
def lambda_handler(event, context):
    try:
        for row in reader:

            # Convert each row to JSON
            record = json.dumps(row)
            print(f"Sending record: {record}")

            # Send record to Kinesis Data Stream
            partition_key = row['Player']
            kinesis_client.put_record(
                StreamName='baseball-stream',
                Data=record,
                PartitionKey=partition_key
            )
        #verify stream complete
        print("Streaming to Kinesis completed.")

    #stream fail and reason for failing
    except Exception as e:
        print(f"Streaming to Kinesis failed: {str(e)}")
```

```
Status: Succeeded
Test Event Name: teststream

Response:
null

Function Logs:
START RequestId: 9f88412b-ce1f-487b-bf63-863aca0c49ee Version: $LATEST
```

Testing Streaming Integration Process

- Reupload baseball_players_march_to_may.csv data in baseball-statistics-dodgers S3 bucket

- Go to CloudWatch and navigate to Log groups under Logs tab
- Under Log groups find /aws/lambda/stream_baseball_to_kinesis to look at lambda streaming records
- In the log streams, click on the latest log stream



/aws/lambda/stream_baseball_to_kinesis

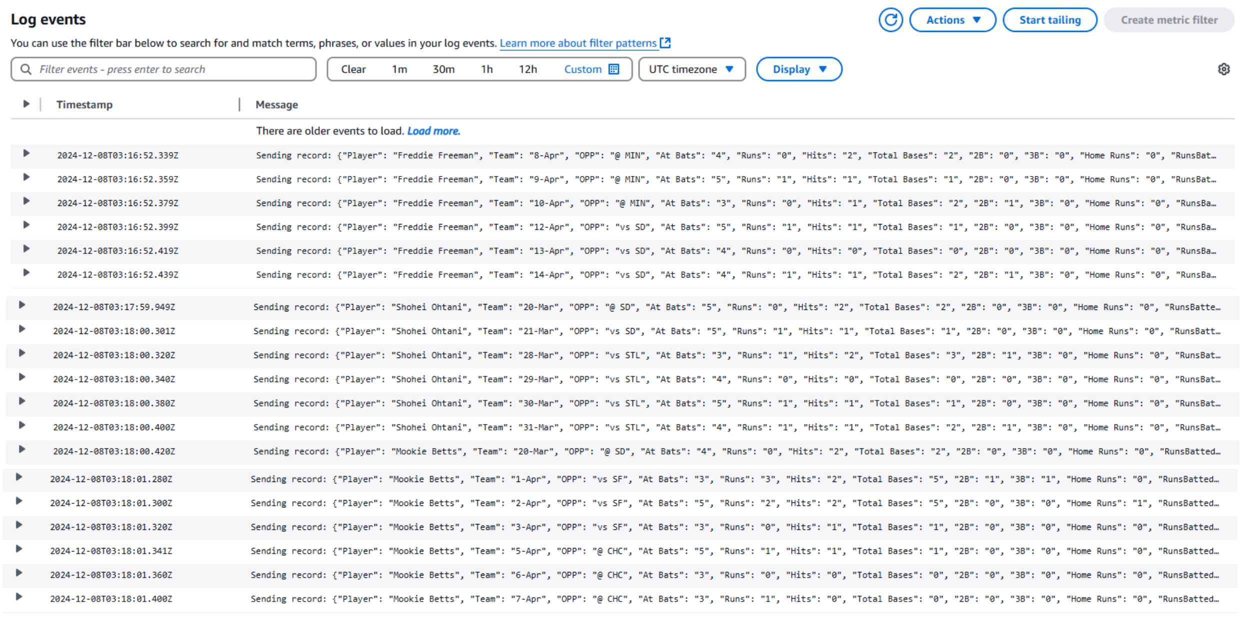
Log group details

Log class: Standard
ARN: arn:aws:logs:us-west-1:124355677069:log-group:/aws/lambda/stream_baseball_to_kinesis:*
Creation time: 8 hours ago
Retention: Never expire
Stored bytes: -
Metric filters: 0
Subscription filters: 0
Contributor Insights rules: -
KMS key ID: -
Anomaly detection: Configure
Data protection: -
Sensitive data count: -
Field indexes: Configure
Transformer: Configure

Log streams (17)

Log stream	Last event time
2024/12/08/[$LATEST]d8e5657966c742fa9c9ee0cc7f3e9d80	2024-12-08 03:16:53 (UTC)

- Check if the stream went successfully. Your log streams should look as following:



Log events

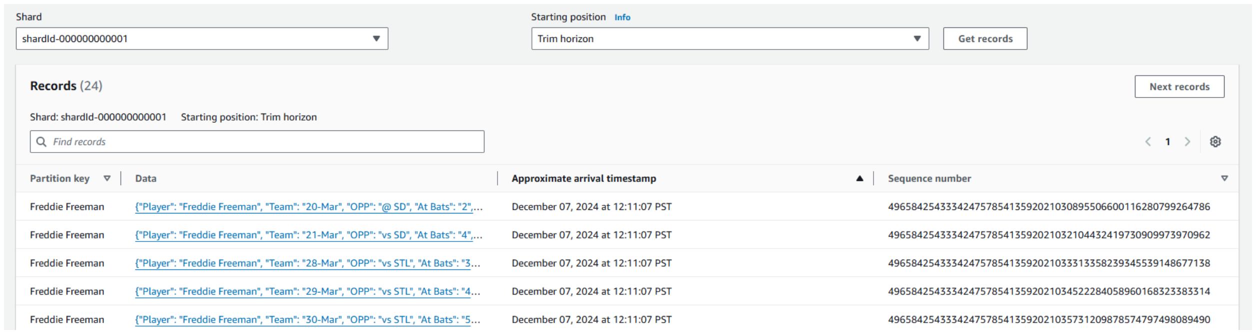
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                   Message
                            There are older events to load. Load more.
2024-12-08T03:16:52.339Z    Sending record: {"Player": "Freddie Freeman", "Team": "8-Apr", "OPP": "@ MIN", "At Bats": "4", "Runs": "0", "Hits": "2", "Total Bases": "2", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBat…
2024-12-08T03:16:52.359Z    Sending record: {"Player": "Freddie Freeman", "Team": "9-Apr", "OPP": "@ MIN", "At Bats": "5", "Runs": "1", "Hits": "1", "Total Bases": "1", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBat…
2024-12-08T03:16:52.379Z    Sending record: {"Player": "Freddie Freeman", "Team": "10-Apr", "OPP": "@ MIN", "At Bats": "3", "Runs": "0", "Hits": "1", "Total Bases": "2", "2B": "1", "3B": "0", "Home Runs": "0", "RunsBa…
2024-12-08T03:16:52.399Z    Sending record: {"Player": "Freddie Freeman", "Team": "12-Apr", "OPP": "vs SD", "At Bats": "5", "Runs": "1", "Hits": "1", "Total Bases": "1", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBa…
2024-12-08T03:16:52.419Z    Sending record: {"Player": "Freddie Freeman", "Team": "13-Apr", "OPP": "vs SD", "At Bats": "4", "Runs": "0", "Hits": "0", "Total Bases": "0", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBa…
2024-12-08T03:16:52.439Z    Sending record: {"Player": "Freddie Freeman", "Team": "14-Apr", "OPP": "vs SD", "At Bats": "4", "Runs": "1", "Hits": "1", "Total Bases": "2", "2B": "1", "3B": "0", "Home Runs": "0", "RunsBa…
2024-12-08T03:17:59.949Z    Sending record: {"Player": "Shohei Ohtani", "Team": "20-Mar", "OPP": "@ SD", "At Bats": "5", "Runs": "0", "Hits": "2", "Total Bases": "2", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBatte…
2024-12-08T03:18:00.301Z    Sending record: {"Player": "Shohei Ohtani", "Team": "21-Mar", "OPP": "vs SD", "At Bats": "5", "Runs": "1", "Hits": "1", "Total Bases": "1", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBatt…
2024-12-08T03:18:00.320Z    Sending record: {"Player": "Shohei Ohtani", "Team": "28-Mar", "OPP": "vs STL", "At Bats": "3", "Runs": "1", "Hits": "2", "Total Bases": "3", "2B": "1", "3B": "0", "Home Runs": "0", "RunsBat…
2024-12-08T03:18:00.340Z    Sending record: {"Player": "Shohei Ohtani", "Team": "29-Mar", "OPP": "vs STL", "At Bats": "4", "Runs": "0", "Hits": "0", "Total Bases": "0", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBat…
2024-12-08T03:18:00.380Z    Sending record: {"Player": "Shohei Ohtani", "Team": "30-Mar", "OPP": "vs STL", "At Bats": "5", "Runs": "1", "Hits": "1", "Total Bases": "1", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBat…
2024-12-08T03:18:00.400Z    Sending record: {"Player": "Shohei Ohtani", "Team": "31-Mar", "OPP": "vs STL", "At Bats": "4", "Runs": "1", "Hits": "1", "Total Bases": "2", "2B": "1", "3B": "0", "Home Runs": "0", "RunsBat…
2024-12-08T03:18:00.420Z    Sending record: {"Player": "Mookie Betts", "Team": "20-Mar", "OPP": "@ SD", "At Bats": "4", "Runs": "0", "Hits": "2", "Total Bases": "2", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBatted…
2024-12-08T03:18:01.280Z    Sending record: {"Player": "Mookie Betts", "Team": "1-Apr", "OPP": "vs SF", "At Bats": "3", "Runs": "3", "Hits": "2", "Total Bases": "5", "2B": "1", "3B": "1", "Home Runs": "0", "RunsBatted…
2024-12-08T03:18:01.300Z    Sending record: {"Player": "Mookie Betts", "Team": "2-Apr", "OPP": "vs SF", "At Bats": "5", "Runs": "2", "Hits": "2", "Total Bases": "5", "2B": "0", "3B": "0", "Home Runs": "1", "RunsBatted…
2024-12-08T03:18:01.320Z    Sending record: {"Player": "Mookie Betts", "Team": "3-Apr", "OPP": "vs SF", "At Bats": "3", "Runs": "0", "Hits": "1", "Total Bases": "1", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBatted…
2024-12-08T03:18:01.341Z    Sending record: {"Player": "Mookie Betts", "Team": "5-Apr", "OPP": "@ CHC", "At Bats": "5", "Runs": "1", "Hits": "1", "Total Bases": "1", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBatted…
2024-12-08T03:18:01.360Z    Sending record: {"Player": "Mookie Betts", "Team": "6-Apr", "OPP": "@ CHC", "At Bats": "3", "Runs": "0", "Hits": "0", "Total Bases": "0", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBatted…
2024-12-08T03:18:01.400Z    Sending record: {"Player": "Mookie Betts", "Team": "7-Apr", "OPP": "@ CHC", "At Bats": "3", "Runs": "1", "Hits": "0", "Total Bases": "0", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBatted…
```

You should be able to see how lambda is streaming to kinesis by organizing according to the baseball player name. This is because we specify in the code that we are going to use the player column as a partition key.

- Go to baseball-stream Kinesis data stream and you can check records coming in to Kinesis under the data viewer tab.
- View Shardid-000000000001 for Freddie Freeman, 000000000002 for Shohei Ohtani, 000000000003 for Mookie Betts

Shardid000000000001



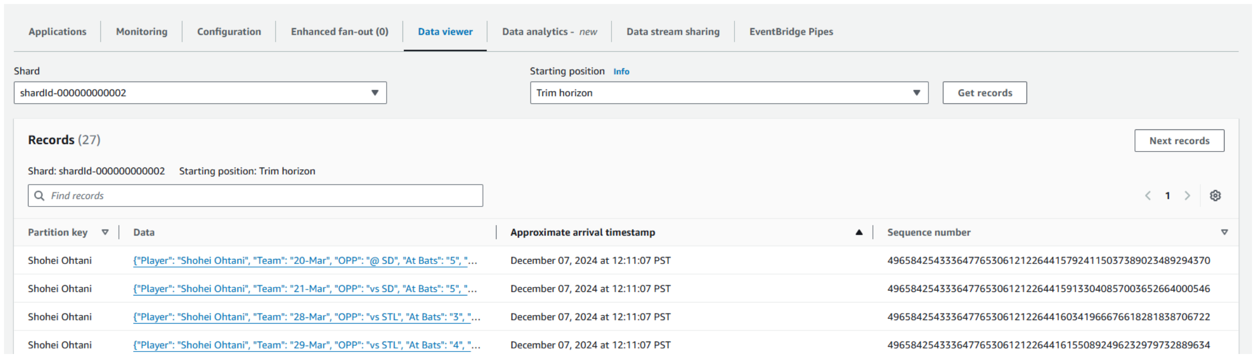
Shard: shardId-000000000001 · Starting position: Trim horizon

Records (24)

Shard: shardId-000000000001 Starting position: Trim horizon

Partition key	Data	Approximate arrival timestamp	Sequence number
Freddie Freeman	{"Player": "Freddie Freeman", "Team": "20-Mar", "OPP": "@ SD", "At Bats": "2",...	December 07, 2024 at 12:11:07 PST	49658425433342475785413592021030895506600116280799264786
Freddie Freeman	{"Player": "Freddie Freeman", "Team": "21-Mar", "OPP": "vs SD", "At Bats": "4",...	December 07, 2024 at 12:11:07 PST	49658425433342475785413592021032104432419730909973970962
Freddie Freeman	{"Player": "Freddie Freeman", "Team": "28-Mar", "OPP": "vs STL", "At Bats": "3...	December 07, 2024 at 12:11:07 PST	49658425433342475785413592021033313358239345539148677138
Freddie Freeman	{"Player": "Freddie Freeman", "Team": "29-Mar", "OPP": "vs STL", "At Bats": "4...	December 07, 2024 at 12:11:07 PST	49658425433342475785413592021034522284058960168323383314
Freddie Freeman	{"Player": "Freddie Freeman", "Team": "30-Mar", "OPP": "vs STL", "At Bats": "5...	December 07, 2024 at 12:11:07 PST	49658425433342475785413592021035731209878574797498089490

Shardid000000000002



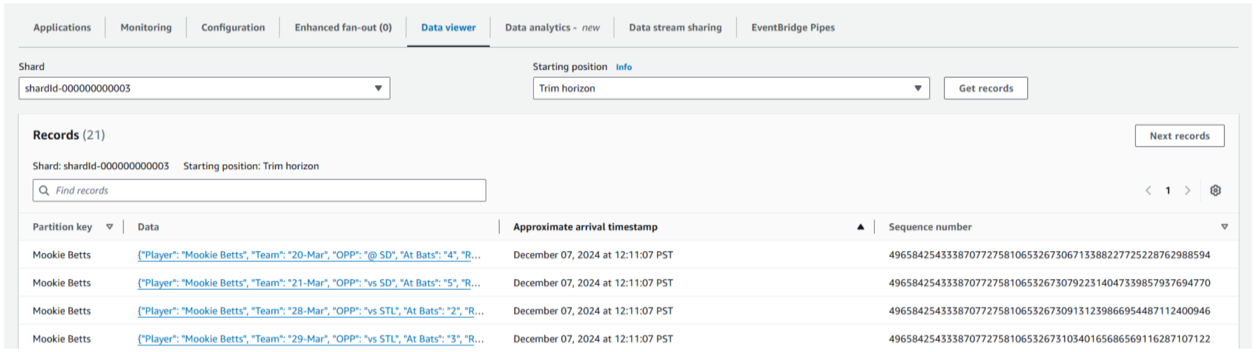
Shard: shardId-000000000002 · Starting position: Trim horizon

Records (27)

Shard: shardId-000000000002 Starting position: Trim horizon

Partition key	Data	Approximate arrival timestamp	Sequence number
Shohei Ohtani	{"Player": "Shohei Ohtani", "Team": "20-Mar", "OPP": "@ SD", "At Bats": "5", "...	December 07, 2024 at 12:11:07 PST	49658425433364776530612122644157924115037389023489294370
Shohei Ohtani	{"Player": "Shohei Ohtani", "Team": "21-Mar", "OPP": "vs SD", "At Bats": "5", "...	December 07, 2024 at 12:11:07 PST	49658425433364776530612122644159133040857003652664000546
Shohei Ohtani	{"Player": "Shohei Ohtani", "Team": "28-Mar", "OPP": "vs STL", "At Bats": "3", "...	December 07, 2024 at 12:11:07 PST	49658425433364776530612122644160341966676618281838706722
Shohei Ohtani	{"Player": "Shohei Ohtani", "Team": "29-Mar", "OPP": "vs STL", "At Bats": "4", "...	December 07, 2024 at 12:11:07 PST	49658425433364776530612122644161550892496232979732889634

Shardid000000000003



Shard: shardId-000000000003 · Starting position: Trim horizon

Records (21)

Shard: shardId-000000000003 Starting position: Trim horizon

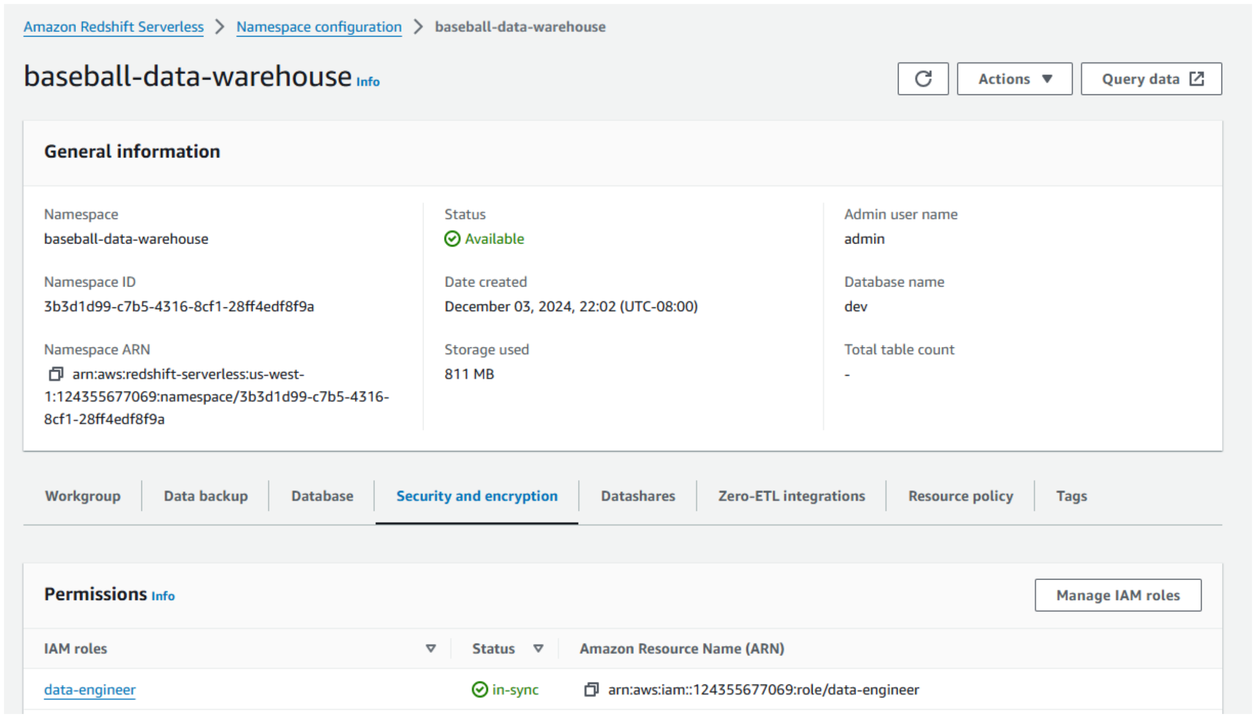
Partition key	Data	Approximate arrival timestamp	Sequence number
Mookie Betts	{"Player": "Mookie Betts", "Team": "20-Mar", "OPP": "@ SD", "At Bats": "4", "R...	December 07, 2024 at 12:11:07 PST	49658425433387077275810653267306713388227725228762988594
Mookie Betts	{"Player": "Mookie Betts", "Team": "21-Mar", "OPP": "vs SD", "At Bats": "5", "R...	December 07, 2024 at 12:11:07 PST	49658425433387077275810653267307922314047339857937694770
Mookie Betts	{"Player": "Mookie Betts", "Team": "28-Mar", "OPP": "vs STL", "At Bats": "2", "R...	December 07, 2024 at 12:11:07 PST	49658425433387077275810653267309131239866954487112400946
Mookie Betts	{"Player": "Mookie Betts", "Team": "29-Mar", "OPP": "vs STL", "At Bats": "3", "R...	December 07, 2024 at 12:11:07 PST	49658425433387077275810653267310340165686569116287107122

As you can see here that each player's statistics are being separated into each shards.

Create a Data Warehouse in Redshift

- Go to Redshift Serverless
- Name target namespace as baseball-data-warehouse. The namespace will host the database
- Name workgroup as workgroupladodgers. Workgroup will host the server
- In VPC choose ITL-vpc that we created
- Choose security group that create as a default with the vpc we created
- Choose private subnets we created for subnet
- Note: there needs to be at least 37 free IP addresses in 2 subnets. Each subnet should be in a different Availability Zone.
- Add data-engineer IAM roles to it.

You should have the following Namespace and workgroup.



Amazon Redshift Serverless > Namespace configuration > baseball-data-warehouse

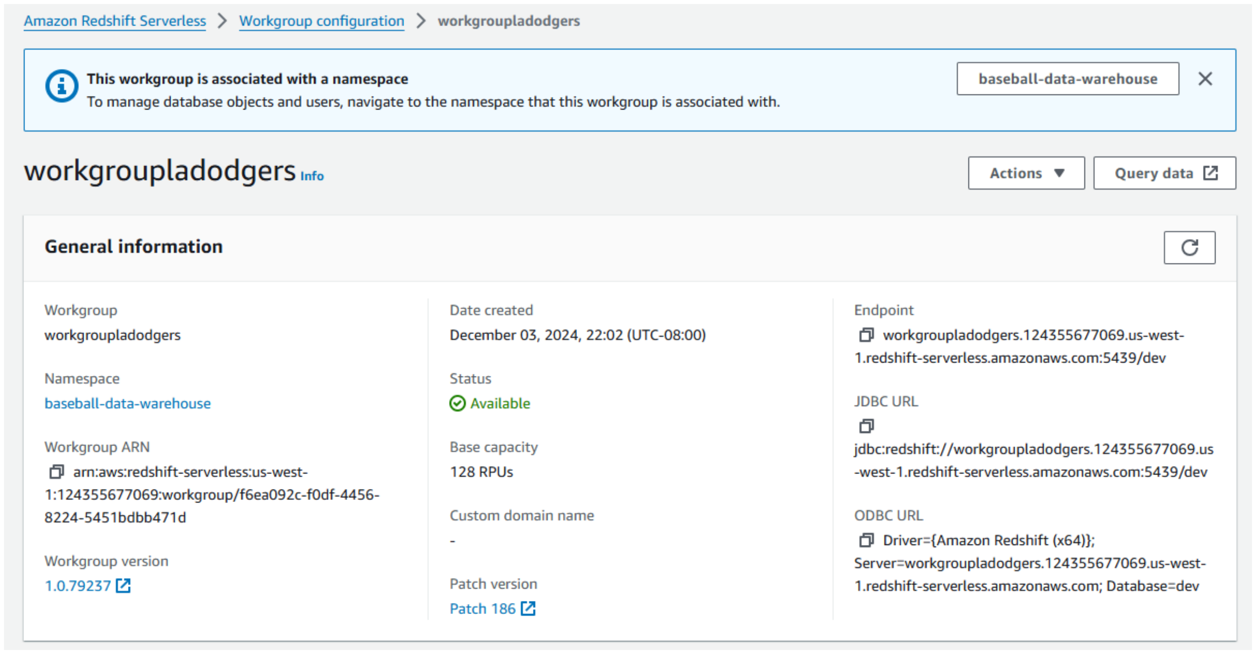
baseball-data-warehouse Info

General information

Namespace baseball-data-warehouse	Status Available	Admin user name admin
Namespace ID 3b3d1d99-c7b5-4316-8cf1-28ff4edf8f9a	Date created December 03, 2024, 22:02 (UTC-08:00)	Database name dev
Namespace ARN arn:aws:redshift-serverless:us-west-1:124355677069:namespace/3b3d1d99-c7b5-4316-8cf1-28ff4edf8f9a	Storage used 811 MB	Total table count -

Permissions Info

IAM roles	Status	Amazon Resource Name (ARN)
data-engineer	in-sync	arn:aws:iam::124355677069:role/data-engineer



Amazon Redshift Serverless > Workgroup configuration > workgroupladodgers

This workgroup is associated with a namespace
To manage database objects and users, navigate to the namespace that this workgroup is associated with.

workgroupladodgers Info

General information

Workgroup workgroupladodgers	Date created December 03, 2024, 22:02 (UTC-08:00)	Endpoint workgroupladodgers.124355677069.us-west-1.redshift-serverless.amazonaws.com:5439/dev
Namespace baseball-data-warehouse	Status Available	JDBC URL jdbc:redshift://workgroupladodgers.124355677069.us-west-1.redshift-serverless.amazonaws.com:5439/dev
Workgroup ARN arn:aws:redshift-serverless:us-west-1:124355677069:workgroup/f6ea092c-f0df-4456-8224-5451bdbb471d	Base capacity 128 RPUs	ODBC URL Driver={Amazon Redshift (x64)}; Server=workgroupladodgers.124355677069.us-west-1.redshift-serverless.amazonaws.com; Database=dev
Workgroup version 1.0.79237	Custom domain name -	
	Patch version Patch 186	

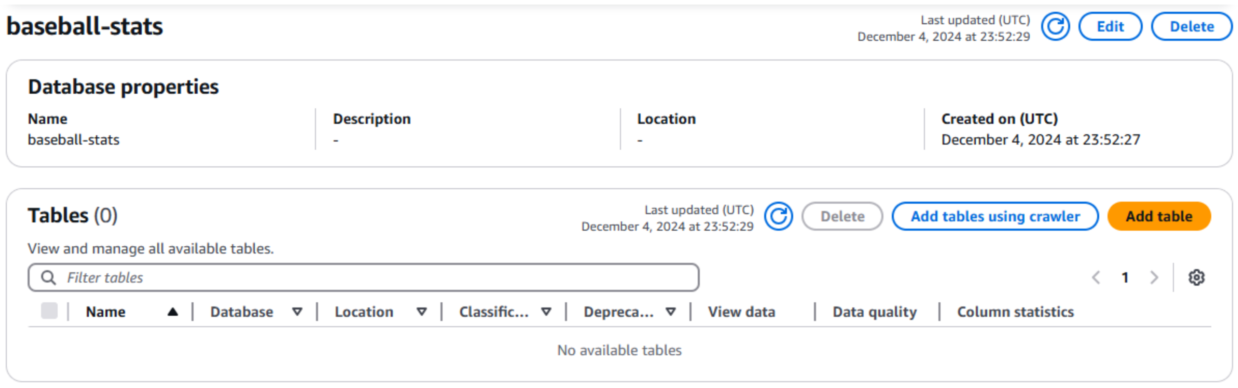
Transforming Data with AWS Glue

Create Database

- Go to Amazon Glue and click on Databases
- Create new database by click add database

- Enter the name of database baseball-stats
- Click create database

Your Database should look as the following:



baseball-stats

Last updated (UTC) December 4, 2024 at 23:52:29

Database properties

Name	Description	Location	Created on (UTC)
baseball-stats	-	-	December 4, 2024 at 23:52:27

Tables (0)

Last updated (UTC) December 4, 2024 at 23:52:29

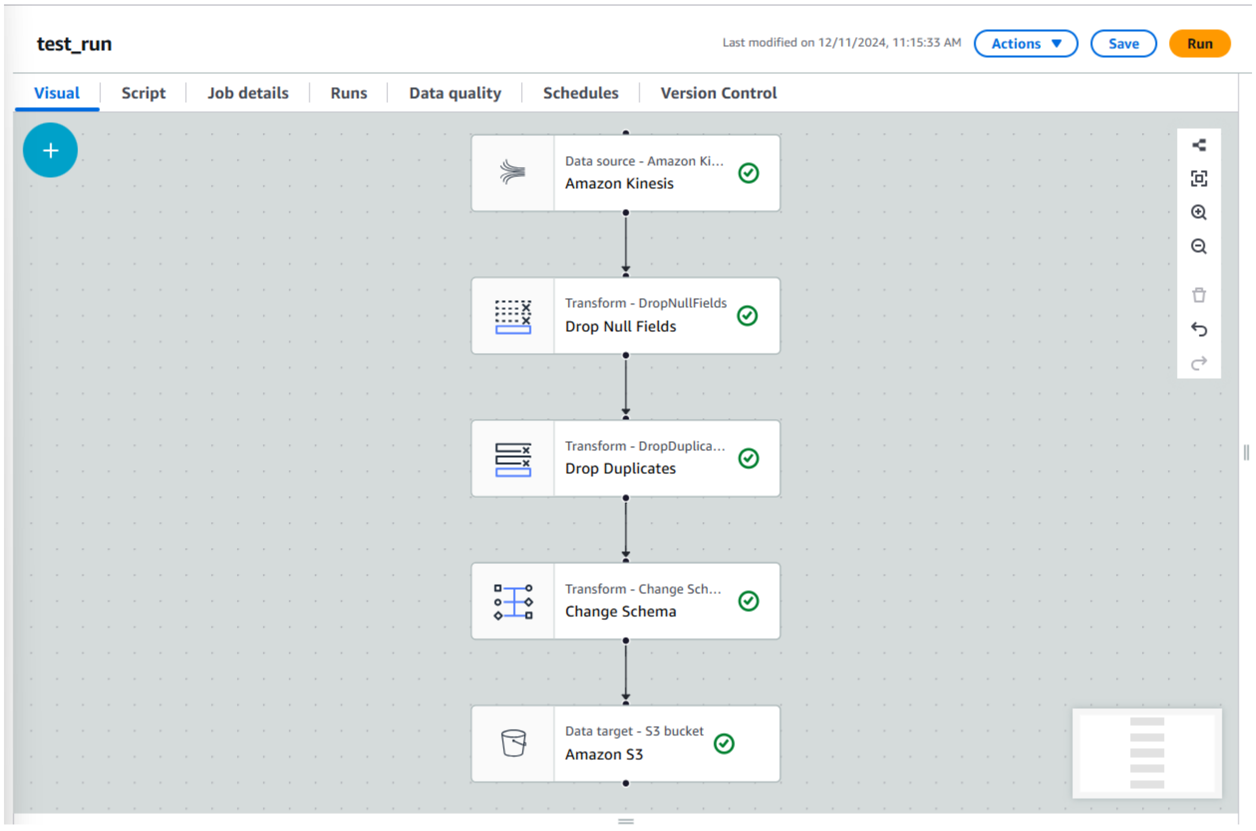
View and manage all available tables.

Name	Database	Location	Classific...	Depreca...	View data	Data quality	Column statistics

No available tables

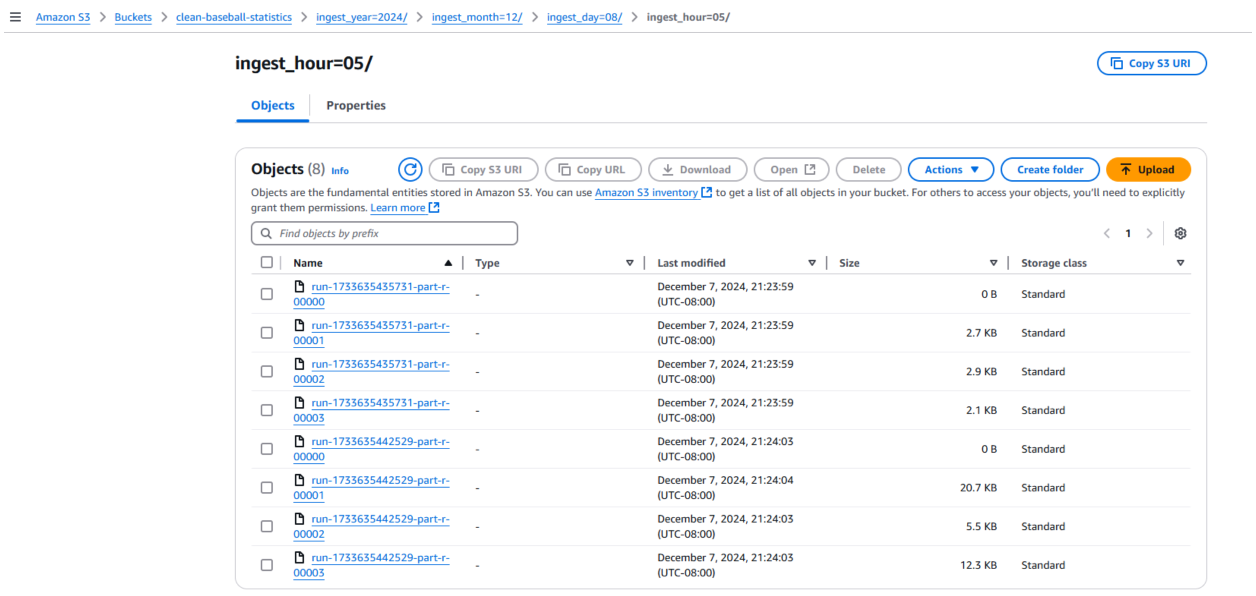
Use Visual ETL to transform data

- Go to AWS Glue and to Visual ETL
- Choose the source as baseball-stream Kinesis stream where we have untransformed data
  - Choose stream details
  - Starting position: Earliest
  - Window size:100
  - Choose dataformat JSON
- Choose Drop Null Fields
  - For dropnullfields, click check for empty string, “null” string, and -1 Integer
- Choose Drop Duplicates
- Choose change schema for transformation
  - keep only player, team change to date, homeruns, runbattedin, avg, and on-base percentage
  - Cast homeruns and runbattedin to integer while cast avg and on-base percentage for double
- Choose target as S3 bucket clean-baseball-statistics to store transformed data
  - Choose JSON format, uncompressed
  - Uncheck the box for data quality because it will failed the run job for not importing aws data quality library.
- Go to job detail tab
  - Make sure the IAM role is data engineer
  - Make sure type is spark streaming
  - Choose worker type G.25X which is suitable for streaming
  - Check automatically scale the number of workers
  - Input maximum number of workers you want to use. I chose 5 workers



Refs:
https://www.youtube.com/watch?v=6ggTFOtfUxU
https://www.youtube.com/watch?v=ziEEdL1egPk

- Check in clean-baseball-statistics S3 bucket
- You should see folder inside called ingest_year=2024/, ingest_month=12/, ingest_day=8/, and hour of ingestion



Amazon S3 > Buckets > clean-baseball-statistics > ingest_year=2024/ > ingest_month=12/ > ingest_day=08/ > ingest_hour=05/

ingest_hour=05/

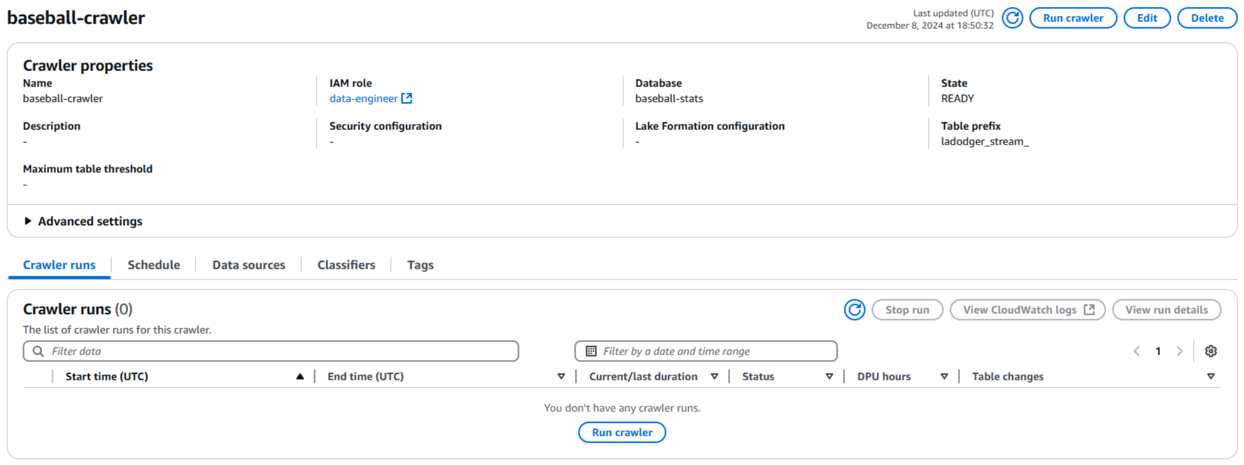
Objects (8)

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
run-1733635435731-part-r-00000	-	December 7, 2024, 21:23:59 (UTC-08:00)	0 B	Standard
run-1733635435731-part-r-00001	-	December 7, 2024, 21:23:59 (UTC-08:00)	2.7 KB	Standard
run-1733635435731-part-r-00002	-	December 7, 2024, 21:23:59 (UTC-08:00)	2.9 KB	Standard
run-1733635435731-part-r-00003	-	December 7, 2024, 21:23:59 (UTC-08:00)	2.1 KB	Standard
run-1733635442529-part-r-00000	-	December 7, 2024, 21:24:03 (UTC-08:00)	0 B	Standard
run-1733635442529-part-r-00001	-	December 7, 2024, 21:24:04 (UTC-08:00)	20.7 KB	Standard
run-1733635442529-part-r-00002	-	December 7, 2024, 21:24:03 (UTC-08:00)	5.5 KB	Standard
run-1733635442529-part-r-00003	-	December 7, 2024, 21:24:03 (UTC-08:00)	12.3 KB	Standard

Use crawler to extract the schema and metadata of transformed data

- Go to AWS Glue and to crawler
- Click create crawler
- Name the crawler baseball-crawler
- Add datasource and choose S3 bucket clean-baseball-statistics
- choose data engineer IAM role
- Choose baseball database
- Choose prefix ladodger_stream_
- Choose on demand for crawler schedule



baseball-crawler

Last updated (UTC) December 8, 2024 at 18:50:32

Crawler properties

Name: baseball-crawler	IAM role: data-engineer	Database: baseball-stats	State: READY
Description: -	Security configuration: -	Lake Formation configuration: -	Table prefix: ladodger_stream_
Maximum table threshold: -			

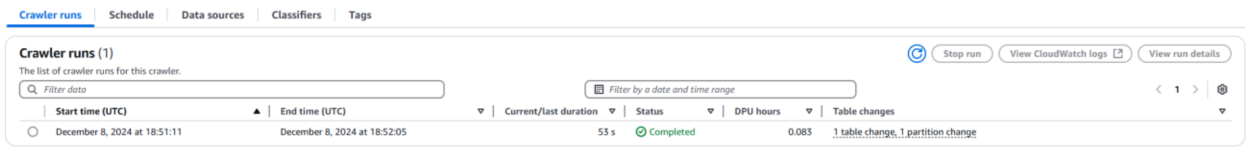
Crawler runs (0)

The list of crawler runs for this crawler.

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes

You don't have any crawler runs.

After that you should be able to see a table in the baseball-stream database once the crawler is done.

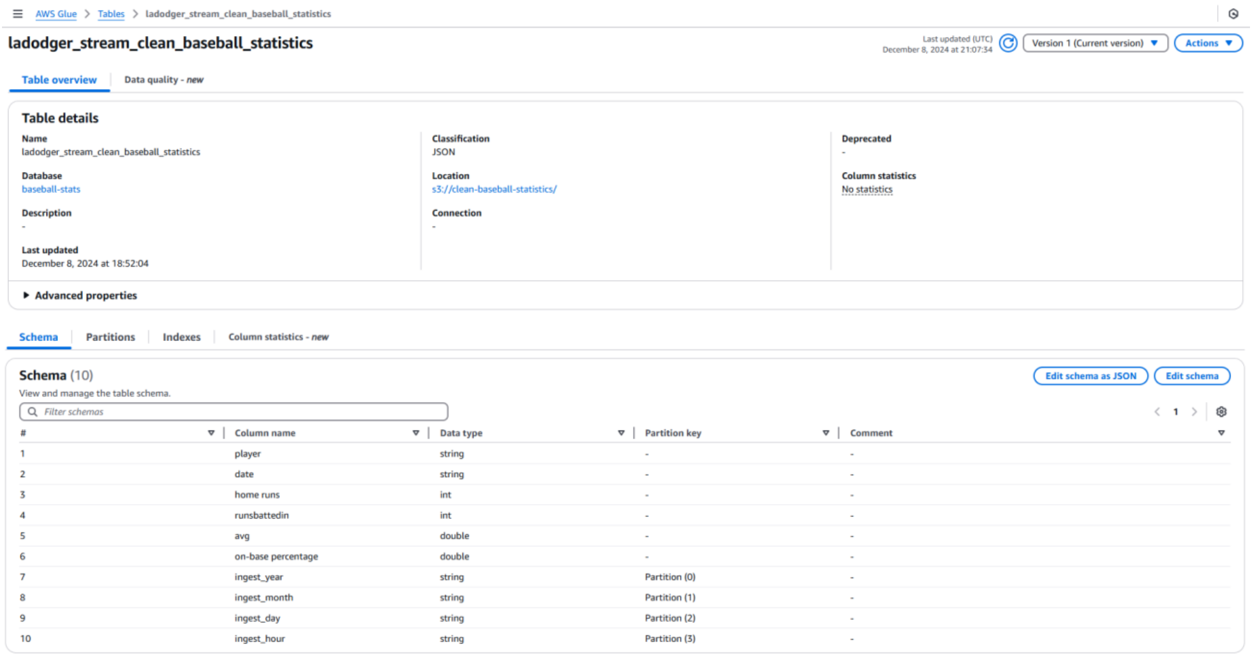


Crawler runs (1)

The list of crawler runs for this crawler.

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
December 8, 2024 at 18:51:11	December 8, 2024 at 18:52:05	53 s	Completed	0.083	1 table change, 1 partition change

Go to database baseball-stats and to ladodger_stream_clean_baseball_statistics that we made



ladodger_stream_clean_baseball_statistics

Last updated (UTC) December 8, 2024 at 21:07:34

Table details

Name: ladodger_stream_clean_baseball_statistics	Classification: JSON	Deprecated: -
Database: baseball-stats	Location: s3://clean-baseball-statistics/	Column statistics: No statistics
Description: -	Connection: -	
Last updated: December 8, 2024 at 18:52:04		

Schema (10)

View and manage the table schema.

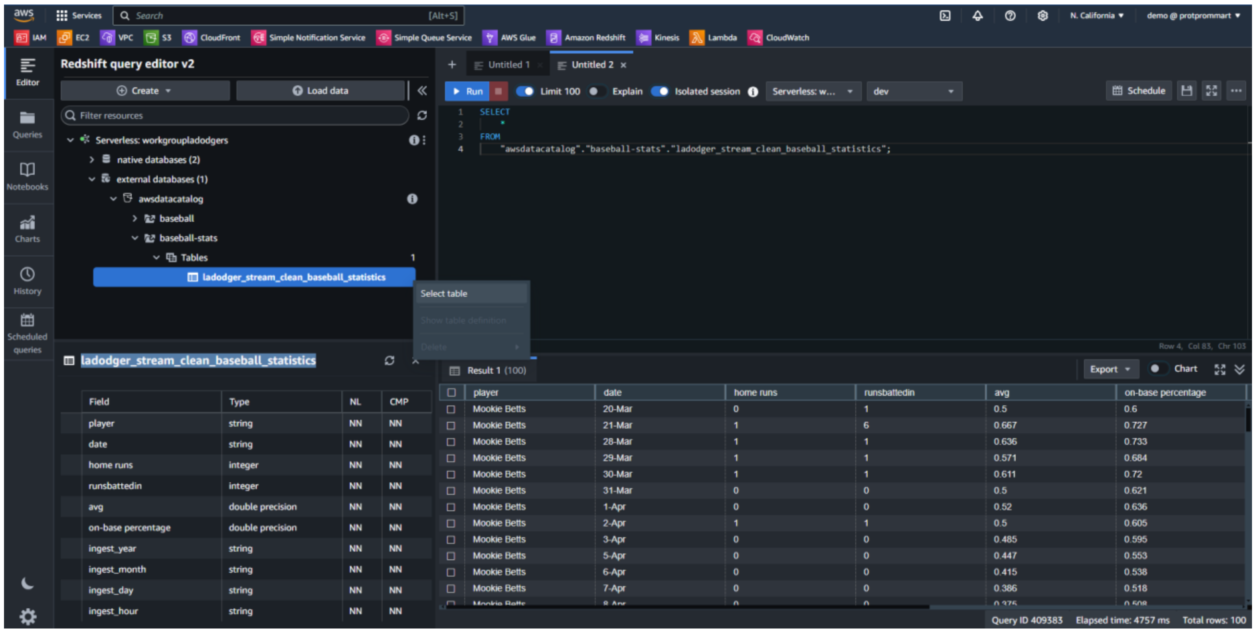
#	Column name	Data type	Partition key	Comment
1	player	string	-	-
2	date	string	-	-
3	home runs	int	-	-
4	runsbattedin	int	-	-
5	avg	double	-	-
6	on-base percentage	double	-	-
7	ingest_year	string	Partition (0)	-
8	ingest_month	string	Partition (1)	-
9	ingest_day	string	Partition (2)	-
10	ingest_hour	string	Partition (3)	-

We see that ingest year, month, day, and hour column were added to save when records were brought in by Kinesis

Use Redshift Query Editor to visualize and query transformed data

Query Transformed Data

- Go to workgroup on redshift
- Click on query data
- In redshift query editor go to external databases, and into baseball-stats database, and into the ladodger_stream_clean_baseball_statistics table created by crawler
- Right click on the table, choose select table


```
SELECT
    *
FROM
    "awsdatacatalog"."baseball-stats"."ladodger_stream_clean_baseball_statistics";
```

ladodger_stream_clean_baseball_statistics

Field	Type	NL	CMP
player	string	NN	NN
date	string	NN	NN
home runs	integer	NN	NN
runsbattedin	integer	NN	NN
avg	double precision	NN	NN
on-base percentage	double precision	NN	NN
ingest_year	string	NN	NN
ingest_month	string	NN	NN
ingest_day	string	NN	NN
ingest_hour	string	NN	NN

Result 1 (100)

player	date	home runs	runsbattedin	avg	on-base percentage
Mookie Betts	20-Mar	0	1	0.5	0.6
Mookie Betts	21-Mar	1	6	0.667	0.727
Mookie Betts	28-Mar	1	1	0.636	0.733
Mookie Betts	29-Mar	1	1	0.571	0.684
Mookie Betts	30-Mar	1	1	0.611	0.72
Mookie Betts	31-Mar	0	0	0.5	0.621
Mookie Betts	1-Apr	0	0	0.52	0.636
Mookie Betts	2-Apr	1	1	0.5	0.605
Mookie Betts	3-Apr	0	0	0.485	0.595
Mookie Betts	5-Apr	0	0	0.447	0.553
Mookie Betts	6-Apr	0	0	0.415	0.538
Mookie Betts	7-Apr	0	0	0.386	0.518
Mookie Betts	8-Apr	0	0	0.375	0.508

Query ID 409383 Elapsed time: 4757 ms Total rows: 100

Your table should display as this.

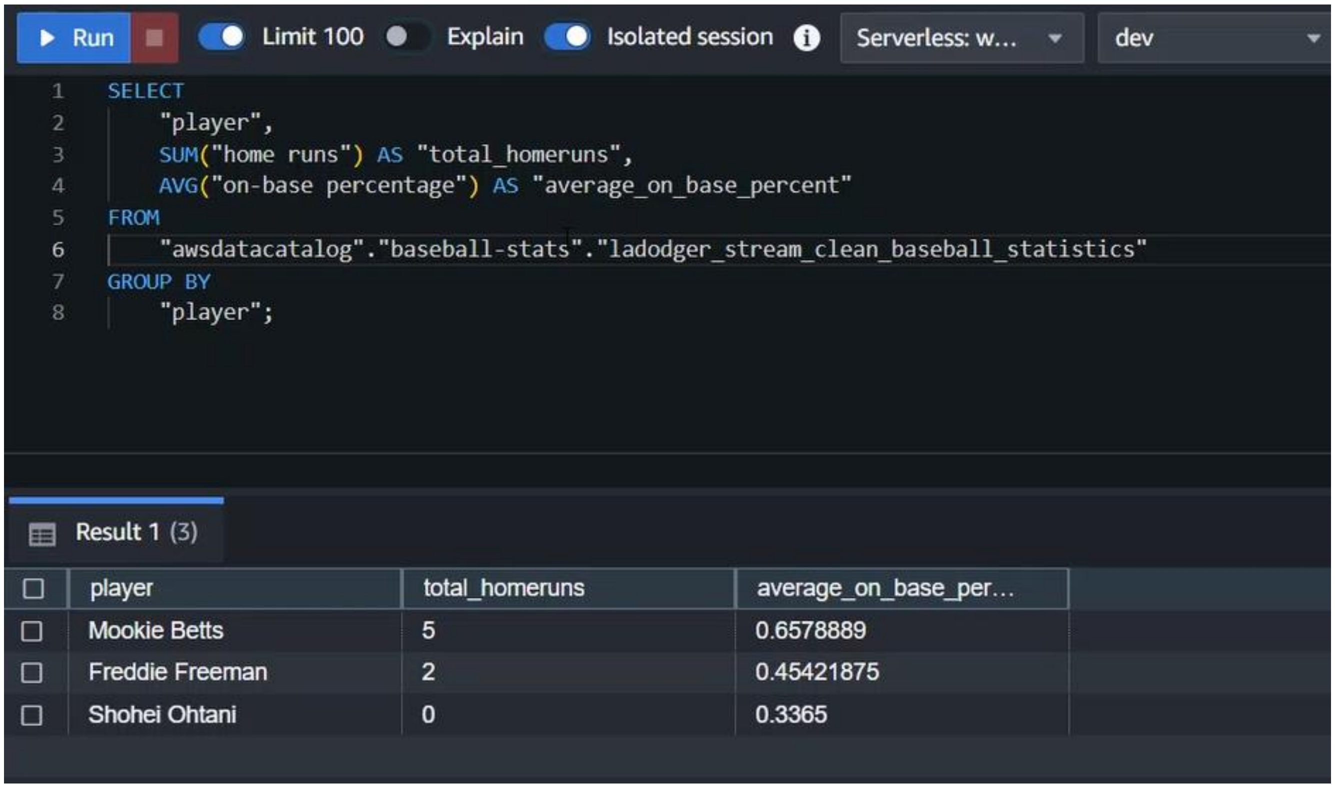
Visualize Data

Our goal in querying and visualizing is, let's say that you have to deliver best players statistics to the company that carries out a sports betting service, for the company to give statistics to users whether they are going to bet on among three batters. We are going to be looking especially at 2 statistics for batters position including an average of, on-base percentage, on average what is the percentage that the player hit the ball and make it safe to a base, and total home runs they made in this season so far.

- Use the following query in redshift query editor to calculate statistics

```
SELECT
    "player",
    SUM("home runs") AS "total_homeruns",
    AVG("on-base percentage") AS "average_on_base_percent"
FROM
    "awsdatacatalog"."baseball-
stats"."ladodger_stream_clean_baseball_statistics"
GROUP BY
    "player";
```

- You should be able to see the following query result

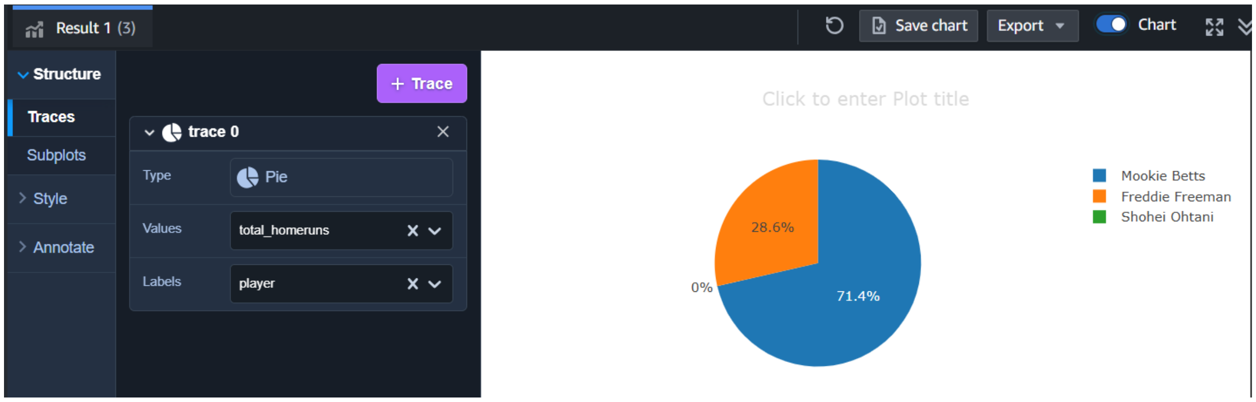


```sql
SELECT
    "player",
    SUM("home runs") AS "total_homeruns",
    AVG("on-base percentage") AS "average_on_base_percent"
FROM
    "awsdatacatalog"."baseball-stats"."ladodger_stream_clean_baseball_statistics"
GROUP BY
    "player";
```

player	total_homeruns	average_on_base_per...
Mookie Betts	5	0.6578889
Freddie Freeman	2	0.45421875
Shohei Ohtani	0	0.3365

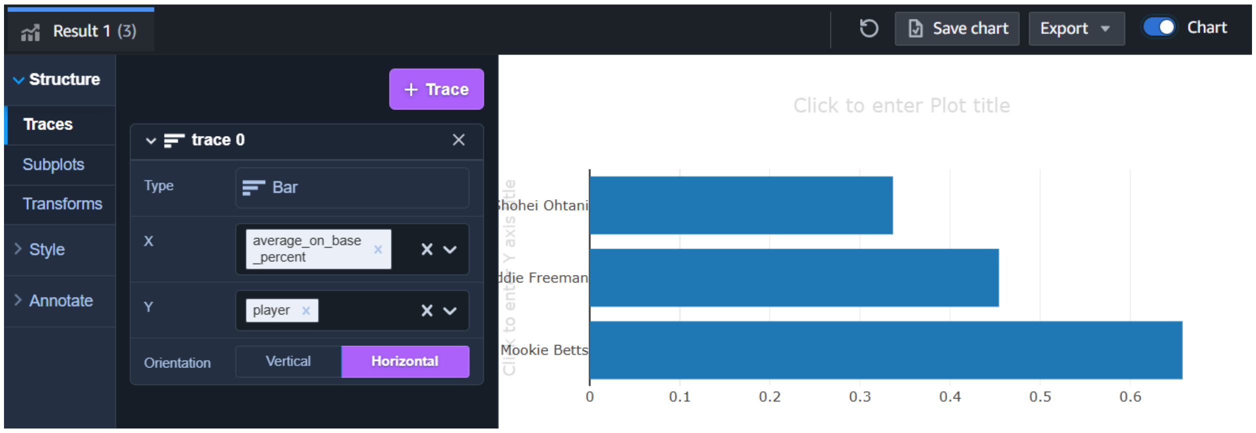
- Click on chart button to view query result in chart
- Choose the following:
    - type: barchart
    - Values: total homeruns
    - Labels: player

- Choose values as average_on_base_percent to view horizontal bar-chart of average on-base percentage
- You can optionally export this chart by click on export button



Type: Pie
Values: total_homeruns
Labels: player

- Based on these statistics, a batter that you should bet on is Mookie Betts because he has the highest total home runs and average on base percentage.



Type: Bar

X: average_on_base_percent

Y: player

Orientation: Vertical [✓ Horizontal]