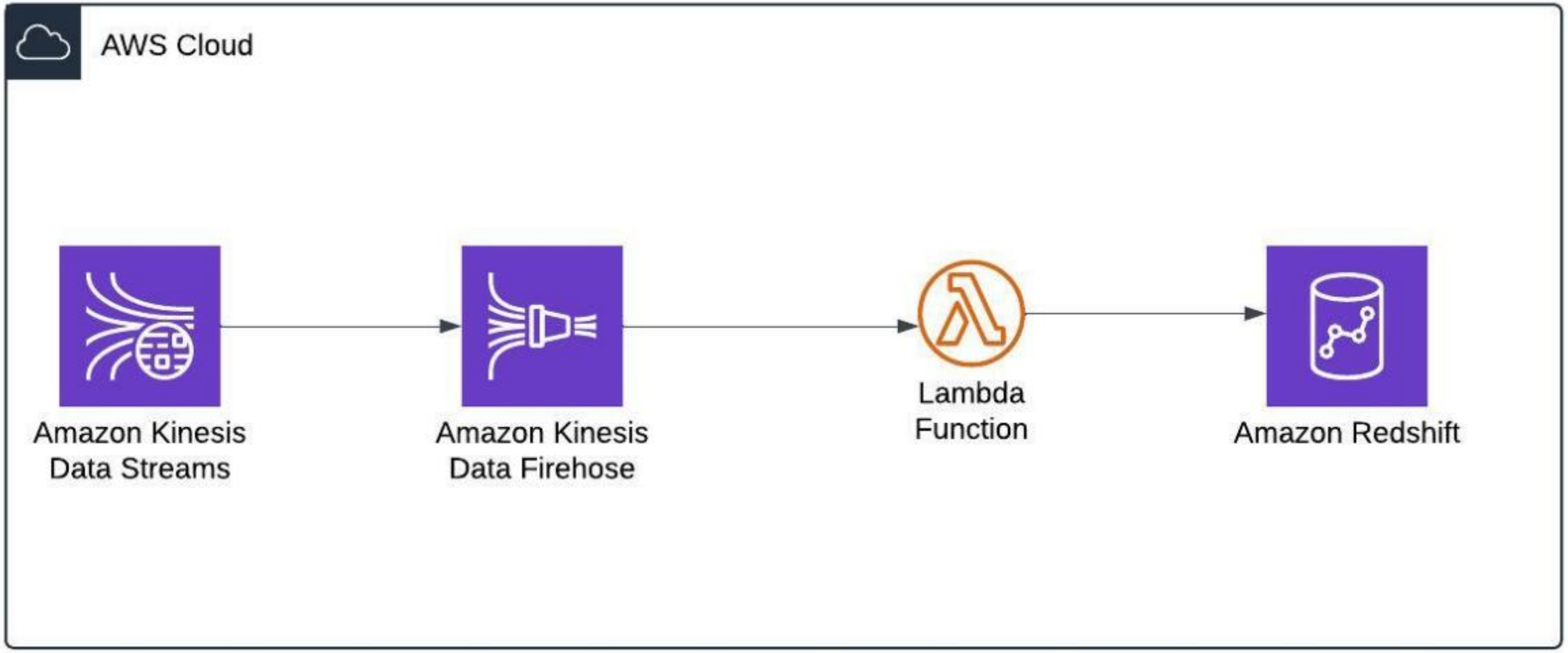
Real-Time Data Pipeline with Kinesis Data Streams, Lambda, Firehose, S3, and Redshift for Data Validation and Ingestion

Objective

- AWS Kinesis Data Streams: Capture live, real-time data from various sources.
- AWS Kinesis Data Firehose: Stream the captured data into Amazon S3 and use AWS Lambda for real-time validation and processing.
- AWS Lambda: Validate and process the incoming data before it is inserted into Amazon Redshift.
- Amazon S3: Temporarily store the data before it undergoes validation and transformation through Lambda.
- Amazon Redshift: Load and store validated and processed data for fast querying and analytics.

Tools Used

- AWS Kinesis Data Streams: Captures live, real-time data from various sources.
- AWS Kinesis Data Firehose: Streams the validated data to Amazon S3 and integrates with Lambda for processing.
- AWS Lambda: Validates and processes the incoming data before it is inserted into Redshift.
- Amazon S3: Serves as a temporary storage location for data before it is validated and ingested into Redshift.
- Amazon Redshift: A data warehouse where processed data is stored and made available for analytics.
- AWS SDK (boto3): Used within Lambda to interact with AWS services, including inserting data into Redshift.


AWS Cloud
Amazon Kinesis Data Streams
Amazon Kinesis Data Firehose
Lambda Function
Amazon Redshift

Amazon Kinesis

Dashboard

Data streams

Amazon Data Firehose New

Managed Apache Flink New

Resources

CloudFormation templates

AWS Glue Schema Registry

Data streams (1) Info

Process data in real time | Create a Firehose stream | Actions | Create data stream

Find data streams

1

Name	Status	Capacity mode	Provisioned shards	Sharing policy	Data retention period	Encryption
app_clickstream_events	Active	On-demand	-	No	1 day	Disabled

Functions (1)

Last fetched 0 seconds ago

Actions

Create function

Filter by attributes or search by keyword

1

Function name	Description	Package type	Runtime	Last modified
clickstream_events_enrichment	-	Zip	Python 3.10	2 hours ago

clickstream_events_enrichment

Throttle | Copy ARN | Actions

Function overview Info

Export to Infrastructure Composer | Download

Diagram | Template

clickstream_events_enrichment

Layers (0)

+ Add trigger

+ Add destination

Description
-

Last modified
2 hours ago

Function ARN
arn:aws:lambda:us-east-1:418272783111:function:clickstream_events_enrichment

Function URL Info
-

Code | Test | Monitor | Configuration | Aliases | Versions

Lambda > Functions > clickstream_events_enrichment

Code source Info

Upload from

clickstream_events_enrichment

EXPLORER

CLICKSTREAM_EVENTS_ENRICHMENT

lambda_function.py

DEPLOY

Deploy (Ctrl+Shift+U)

Test (Ctrl+Shift+I)

TEST EVENTS [NONE SELECTED]

+ Create new test event

lambda_function.py

lambda_function.py

```python
import json
import base64
import logging

# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger()

# Helper function to determine if the browser indicates a mobile device
def is_mobile_browser(browser):
    # List of mobile indicators based on typical mobile browser strings
    mobile_indicators = ['Mobile', 'iPhone', 'Android', 'iPad']
    return any(indicator in browser for indicator in mobile_indicators)

# Helper function to categorize traffic sources
def categorize_traffic_source(source):
    if 'Google' in source or 'Bing' in source:
        return 'Search Engine'
    elif 'Facebook' in source or 'YouTube' in source:
```

lambda.py > categorize_traffic_source

```python
import json
import base64
import logging

# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger()

# Helper function to determine if the browser indicates a mobile device
def is_mobile_browser(browser):
    # List of mobile indicators based on typical mobile browser strings
    mobile_indicators = ['Mobile', 'iPhone', 'Android', 'iPad']
    return any(indicator in browser for indicator in mobile_indicators)

# Helper function to categorize traffic sources
def categorize_traffic_source(source):
    if 'Google' in source or 'Bing' in source:
        return 'Search Engine'
    elif 'Facebook' in source or 'YouTube' in source:
        return 'Social Media'
    elif 'Email' in source:
        return 'Email'
    else:
        return 'Direct'

def lambda_handler(event, context):
    output = []

    # Process each record in the event batch
    for record in event['records']:
        try:
```

lambda.py > categorize_traffic_source

```python

def lambda_handler(event, context):
    output = []

    # Process each record in the event batch
    for record in event['records']:
        try:
            # Decode the incoming data from base64 and convert from JSON
            payload = json.loads(base64.b64decode(record['data']))

            # Extract fields to be included in the CSV
            id = payload.get('id', '')
            user_id = payload.get('user_id', '')
            session_id = payload.get('session_id', '')
            ip_addre
                                    ess', '')
            city = p (variable) payload: Any
            state = payload.get('state', '')
            browser = payload.get('browser', '')
            traffic_source = payload.get('traffic_source', '')
            uri = payload.get('uri', '')
            event_type = payload.get('event_type', '')

            # Derive is_mobile and traffic_source_category
            is_mobile = 'Yes' if is_mobile_browser(browser) else 'No'
            traffic_source_category = categorize_traffic_source(traffic_source)

            # Quote each field to ensure proper CSV formatting
            data = f'"{id}","{user_id}","{session_id}","{ip_address}","{city}","{state}","{browser}","{is_mobile}","{traffic

            # Encode the CSV string to base64
            encoded_csv = base64.b64encode(data.encode('utf-8')).decode('utf-8')
```

lambda.py > categorize_traffic_source

```python
def lambda_handler(event, context):
            data = f'"{id}","{user_id}","{session_id}","{ip_address}","{city}","{state}","{browser}","{is_mobile}","{traffic

            # Encode the CSV string to base64
            encoded_csv = base64.b64encode(data.encode('utf-8')).decode('utf-8')

            # Prepare the output record with unique recordId and transformed data
            output_record = {
                'recordId': record['recordId'],
                'result': 'Ok',
                'data': encoded_csv
            }
            output.append(output_record)

            logger.info(f"Processed record ID: {record['recordId']}")

        except Exception as e:
            logger.error(f"Error processing record {record['recordId']}: {str(e)}")
            # Return the original record in case of failure
            output_record = {
                'recordId': record['recordId'],
                'result': 'ProcessingFailed',
                'data': record['data']  # Return original data
            }
            output.append(output_record)

    # Return the transformed records to Firehose
    logger.info(f"Total records processed: {len(output)}")
    return {'records': output}
```

Services | Search [Alt+S] | United States (N. Virginia) | manishtripathi

Redshift query editor v2

Create | Load data

Filter resources

Serverless: default-workgr...

Untitled 1 | Untitled 2 | Untitled 3 | Untitled 4 | Untitled 5 | Untitled 6

Run | Limit 100 | Explain | Isolated session | Serverless: de... | clickstream_db

Schedule

```sql
create database clickstream_db;

CREATE TABLE prod_schema.events (
    id BIGINT,
    user_id BIGINT,
    session_id CHAR(36),
    ip_address VARCHAR(50),
    city VARCHAR(255),
    state VARCHAR(255),
    browser VARCHAR(50),
    is_mobile VARCHAR(3),
    traffic_source VARCHAR(50),
    traffic_source_category VARCHAR(50),
    uri VARCHAR(255),
    event_type VARCHAR(50)
);

SELECT * from prod_schema.events;
```

Row 3, Col 14, Chr

Result 1 | Export | Chart

Amazon S3 > Buckets

General purpose buckets | Directory buckets

General purpose buckets (5) Info All AWS Regions

Buckets are containers for data stored in S3.

Copy ARN | Empty | Delete | Create bucket

Find buckets by name

1

Name	AWS Region	IAM Access Analyzer	Creation date
aws-glue-assets-418272783111-us-east-1	US East (N. Virginia) us-east-1	View analyzer for us-east-1	December 8, 2024, 13:06:32 (UTC+05:30)
aws-logs-418272783111-us-east-1	US East (N. Virginia) us-east-1	View analyzer for us-east-1	December 10, 2024, 08:46:40 (UTC+05:30)
gd-airflow	US East (N. Virginia) us-east-1	View analyzer for us-east-1	December 20, 2024, 11:10:44 (UTC+05:30)
gd-aws-de-labs	US East (N. Virginia) us-east-1	View analyzer for us-east-1	December 20, 2024, 22:48:51 (UTC+05:30)
testing65987	US East (N. Virginia) us-east-1	View analyzer for us-east-1	December 8, 2024, 13:01:11 (UTC+05:30)

Amazon Data Firehose > Firehose streams

Amazon Data Firehose

Firehose streams

Resources

What's new

Developer guide

API reference

Firehose streams (1) Info

Delete | Create Firehose stream

You can create a Firehose stream to set up a source, destination, and optional transformation for your streaming data delivery.

Find Firehose streams

1

Name	Status	Creation...	Source	Data tra...	Destinat...	Destinat...
KDS-RED-b...	Active	January 05...	app_clickst...	clickstream...	Amazon Re...	default-w...

Amazon Data Firehose

Firehose streams

Resources

What's new

Developer guide

API reference

Transform records

Edit

Configure Amazon Data Firehose to transform your record data.

AWS Lambda function timeout
The current timeout of the specified AWS Lambda function is 3 seconds. To reduce the risk of the AWS Lambda function timing out before data transformation is complete, increase the timeout to 1 minute or longer in the Advanced settings section of your AWS Lambda configuration. Go to AWS Lambda configuration.

Transform source records with AWS Lambda
Info
On

Buffer size
0.2 MiB

Buffer interval
20 seconds

Lambda function
clickstream_events_enrichment

Lambda function version
$LATEST

Description
-

Runtime
python3.10

Timeout
3 seconds

Destination settings Info

Edit

Specify the destination settings for your Firehose stream.

Destination settings Info

Edit

Specify the destination settings for your Firehose stream.

Ensure that your Amazon Redshift Serverless workgroup is publicly accessible and allows inbound access from this Amazon Data Firehose IP address: **52.70.63.192/27**. For more information, see VPC Access to an Amazon Redshift Serverless workgroup.

If you specify Amazon Redshift as your Firehose stream destination, once your Firehose stream is created, you cannot update the specified Amazon Redshift destination type.

Amazon Redshift destination

Serverless workgroup default-workgroup	Database clickstream_db	Columns -
Destination type Serverless workgroup	Table prod_schema.events	User name admin

Amazon Redshift Serverless COPY command Info

COPY command options

CSV IGNOREHEADER 1

COPY command

Amazon Redshift Serverless COPY command Info

COPY command options

CSV IGNOREHEADER 1

COPY command

```
COPY prod_schema.events FROM 's3://gd-aws-de-labs/<manifest>' CREDENTIALS
'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>' MANIFEST CSV
IGNOREHEADER 1 ;
```

Copy

Retry duration

3600 seconds

Intermediate S3 destination

S3 bucket

gd-aws-de-labs

S3 bucket prefix

-

Buffer hints

Buffer size

5 MiB

Compression and encryption

Compression for data records

Not enabled

S3 bucket
gd-aws-de-labs

S3 bucket prefix
-

Buffer hints

Buffer size
5 MiB

Buffer interval
300 seconds

Compression and encryption

Compression for data records
Not enabled

Encryption for data records
Not enabled

Backup settings Info

Edit

Enabling source record backup ensures that source records can be recovered if record processing transformation does not produce the desired results.

Source record backup in Amazon S3
Not enabled

Server-side encryption (SSE) Info

Edit

You can use AWS Key Management Service (KMS) to create and manage keys and to control the use of encryption across a wide range of AWS

gd-aws-de-labs Info

Objects | Metadata - *Preview* | Properties | Permissions | Metrics | Management | Access Points

Objects (5) Info

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

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

Find objects by prefix

1

Name	Type	Last modified	Size	Storage class
2025/	Folder	-	-	-
glue-scripts/	Folder	-	-	-
manifests/	Folder	-	-	-
spotify_data/	Folder	-	-	-
streams/	Folder	-	-	-

Redshift query editor v2

Editor | Queries | Notebooks | Charts | History | Scheduled queries

Create | Load data

Filter resources

Serverless: default-workgr...

Untitled 1 | Untitled 2 | Untitled 3 | Untitled 4 | Untitled 5 | Untitled 6

Run | Limit 100 | Explain | Isolated session | Serverless: de... | clickstream_db

Schedule

```
    is_mobile VARCHAR(3),
    traffic_source VARCHAR(50),
    traffic_source_category VARCHAR(50),
    uri VARCHAR(255),
    event_type VARCHAR(50)
);

SELECT * FROM prod_schema.events LIMIT 10;
```

Row 18, Col 1, Chr 416

Result 1 (10) | Export | Chart

id	user_id	session_id	ip_address	city
1822041	48474	54d4fa70-4baa-4b3f-a5b2...	215.107.45.172	Tianjin
1481573	73714	ffbed0dc-8f35-47b4-a229-...	51.228.47.204	Quanz
870743	28796	5f4f29ec-7aeb-4020-b5c3...	36.6.247.40	Caieira
239679	86653	3c82c565-91dc-4fa0-b90...	144.178.157.62	Tokyo
956291	52902	d1a0168b-bea2-4165-b73...	219.60.17.51	East P
1613231	45421	9056a8f8-e41a-4978-8f44...	158.108.8.144	Mi

Query ID 2443318 | Elapsed time: 76 ms | Total rows: 10