


Multimodal Reasoning AI Agent

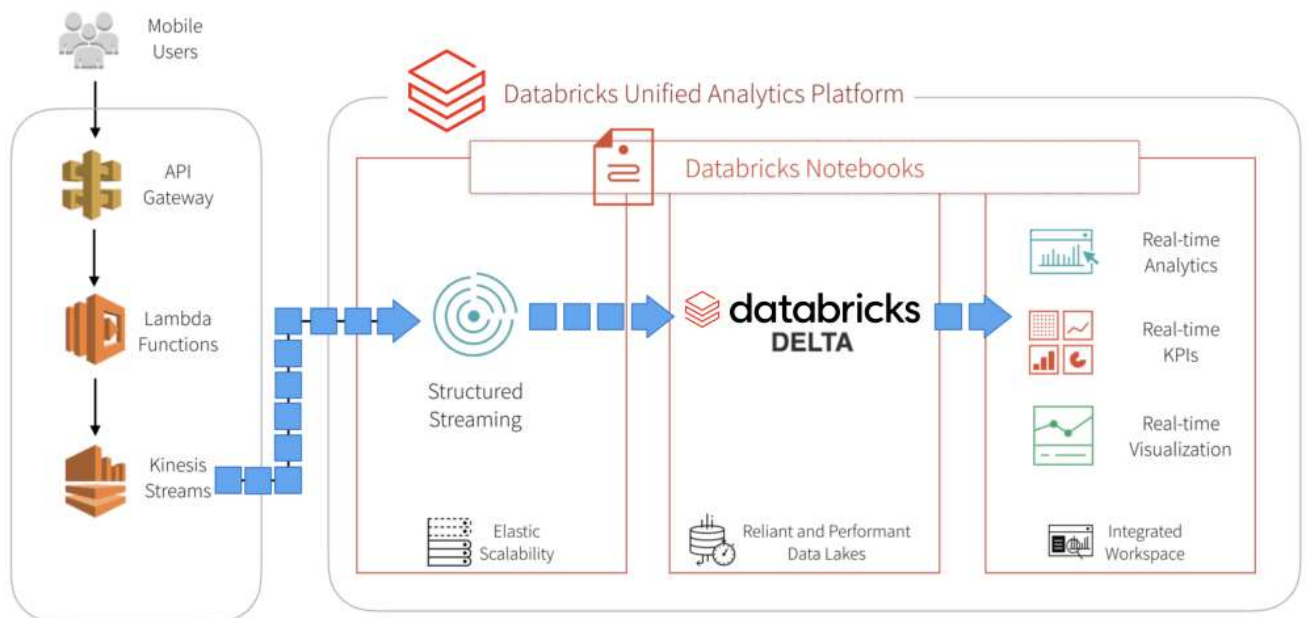
Upload an image and provide a reasoning-based task for the AI Agent. The AI Agent will analyze the image and respond based on your input.

Upload Image

 Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Browse files

 Mobile-Gaming-Events-Data-Pipeline-Databricks.png 30.7KB ×



Uploaded Image

Enter your task/question for the AI Agent:

generate the code for this flow

Analyze Image

AI Response:

Here's a breakdown of the code components involved in the flow you described, along with examples using Python for AWS services and PySpark for Databricks:

Code for the Data Flow

This outlines the code components involved in each stage of the data flow.

1. Mobile Users -> API Gateway

```
*  **Concept:**  Mobile applications interact with the backend through API calls.
*  **Code (Conceptual - This is handled by the mobile app and API Gateway config
...

// Example of an API call from a mobile app (Conceptual)
import requests

api_endpoint = "https://your-api-gateway-endpoint.amazonaws.com/your-stage/you
data = {"event_type": "user_action", "user_id": 123, "timestamp": "2023-10-27T
headers = {'Content-type': 'application/json'}

response = requests.post(api_endpoint, json=data, headers=headers)

if response.status_code == 200:
    print("API call successful")
else:
    print(f"API call failed with status code: {response.status_code}")
...
```

2. API Gateway -> Lambda Functions

```
*  **Concept:** API Gateway triggers a Lambda function upon receiving an API requ
*  **Code (Python - AWS Lambda Function):**

```python
import json
import boto3

kinesis = boto3.client('kinesis')
stream_name = 'your-kinesis-stream-name'

def lambda_handler(event, context):
 try:
 # Extract data from the API Gateway event
 request_body = json.loads(event['body'])

 # Process the data if needed (e.g., validation, transformation)
 # ...

 # Send the data to Kinesis
 response = kinesis.put_record(
 StreamName=stream_name,
```

```

 Data=json.dumps(request_body),
 PartitionKey='partition-key' # Choose an appropriate partition key
)

 return {
 'statusCode': 200,
 'body': json.dumps({'message': 'Data sent to Kinesis'})
 }
except Exception as e:
 print(f"Error processing event: {e}")
 return {
 'statusCode': 500,
 'body': json.dumps({'error': str(e)})
 }
...

```

### ### 3. Lambda Functions -> Kinesis Streams

- \* **Concept:** The Lambda function writes data events to a Kinesis stream.
- \* **Code** (Already included in the Lambda function above):

```

```python
# ... inside the lambda_handler function ...
response = kinesis.put_record(
    StreamName=stream_name,
    Data=json.dumps(request_body),
    PartitionKey='partition-key'
)
...

```

4. Kinesis Streams -> Structured Streaming (Databricks)

- * **Concept:** A Databricks notebook using Structured Streaming reads data from
- * **Code** (PySpark - Databricks Notebook):

```

```python
from pyspark.sql.functions import *
from pyspark.sql.types import StructType, StructField, StringType, LongType, T

Configure Kinesis connection
kinesis_options = {
 "streamName": "your-kinesis-stream-name",
 "endpointUrl": "your-kinesis-endpoint-url", # e.g., 'kinesis.us-east-1.amazonaws.com'
 "startingPosition": "LATEST", # Or "TRIM_HORIZON" to read from the beginning
 "region": "your-aws-region" # e.g., 'us-east-1'
}

Define the schema of your data
schema = StructType([

```

```

 StructField("event_type", StringType(), True),
 StructField("user_id", LongType(), True),
 StructField("timestamp", StringType(), True) # Or TimestampType if you co
 # Add other fields as needed
])

Read data from Kinesis
kinesis_df = spark.readStream.format("kinesis") \
 .options(**kinesis_options) \
 .load()

Deserialize the JSON data
data_df = kinesis_df.selectExpr("CAST(data AS STRING)") \
 .select(from_json("data", schema).alias("payload")) \
 .select("payload.*")

Process the streaming data (e.g., basic transformations)
processed_df = data_df.withColumn("ingestion_time", current_timestamp())

Display the streaming data (for testing)
query = processed_df.writeStream.outputMode("append").format("console").star

Write the processed data to a Delta table (see next step)
...

```

### ### 5. Structured Streaming -> Databricks Delta

```

* **Concept:** The processed streaming data is written to a Delta Lake table in
* **Code (PySpark – Databricks Notebook, continuing from the previous step):**

```python
# ... (previous code for reading from Kinesis) ...

# Define the Delta table path
delta_table_path = "/mnt/data/your_delta_table"

# Write the streaming data to a Delta table
query = processed_df.writeStream \
    .format("delta") \
    .outputMode("append") \
    .option("checkpointLocation", "/mnt/checkpoint/your_delta_table") # Import
    .trigger(processingTime='10 seconds') # Process data in micro-batches
    .start(delta_table_path)

# To read the Delta table later:
# delta_df = spark.read.format("delta").load(delta_table_path)
# delta_df.show()
...

```

6. Databricks Delta -> Real-time Analytics/KPIs/Visualization/Integrated Works

* **Concept:** Data stored in the Delta table is used for real-time analysis, ge
* **Code (PySpark - Databricks Notebook):**

****Real-time Analytics & KPIs:****

```
```python
```

```
Read the Delta table
```

```
delta_table = spark.read.format("delta").load(delta_table_path)
```

```
Example: Calculate a simple KPI (e.g., count of events per type)
```

```
from pyspark.sql.functions import count
```

```
kpi_df = delta_table.groupBy("event_type").agg(count("*").alias("event_count"))
```

```
kpi_df.show()
```

```
For real-time dashboards, you might use Databricks SQL Analytics or connect
...
```

**\*\*Real-time Visualization:\*\***

```
```python
```

```
# You can use libraries like matplotlib or plotly within a Databricks notebook
```

```
# Or connect to a visualization tool like Tableau or Power BI
```

```
import matplotlib.pyplot as plt
```

```
import pandas as pd
```

```
kpi_pandas_df = kpi_df.toPandas()
```

```
plt.bar(kpi_pandas_df['event_type'], kpi_pandas_df['event_count'])
```

```
plt.xlabel("Event Type")
```

```
plt.ylabel("Event Count")
```

```
plt.title("Real-time Event Counts")
```

```
plt.show()
```

```
...
```

****Integrated Workspace:****

* Databricks provides a collaborative environment where you can write and ex

****Important Considerations:****

* ****Error Handling:**** The provided code snippets have basic error handling. In p

* ****Security:**** Ensure proper security configurations for your AWS resources (IA

* ****Scalability and Performance:**** Configure Kinesis Streams, Lambda functions,

* ****Checkpointing:**** Checkpointing in Structured Streaming is crucial for fault

* ****Partitioning:**** Choose appropriate partition keys for Kinesis Streams and D

* ****Schema Evolution:**** Consider how your schema might change over time and how

This comprehensive breakdown and code examples should give you a good starting poi