

Updating Dynamic Data in Place

Lab overview and objectives

A challenge with streaming data from Internet of Things (IoT) devices is that the schema changes frequently. For example, a sensor might send four values at first, then two values, and then ten. With many sensors, this can increase manifold. It can be a challenge to adapt to the ever-changing schema and to update the data lake with changed records. This is particularly a challenge for data lakes that are based on object storage such as in Amazon Simple Storage Service (Amazon S3). The Apache Hudi Connector, which is an open-source tool from AWS Marketplace, can help you to address this challenge.

In this lab, you will use Amazon S3, Amazon Athena, AWS Glue, and Apache Hudi to address the challenge of accommodating a dynamically changing schema. You will use these services to facilitate efficient in-place data updates and run queries to get data in near real time.

After completing this lab, you should be able to do the following:

- Create an AWS Glue job to run custom extract, transform, and load (ETL) scripts.
- Use Athena to run queries.
- Use the Apache Hudi Connector to perform in-place updates.

Duration

This lab will require approximately **90 minutes** to complete.

AWS service restrictions

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

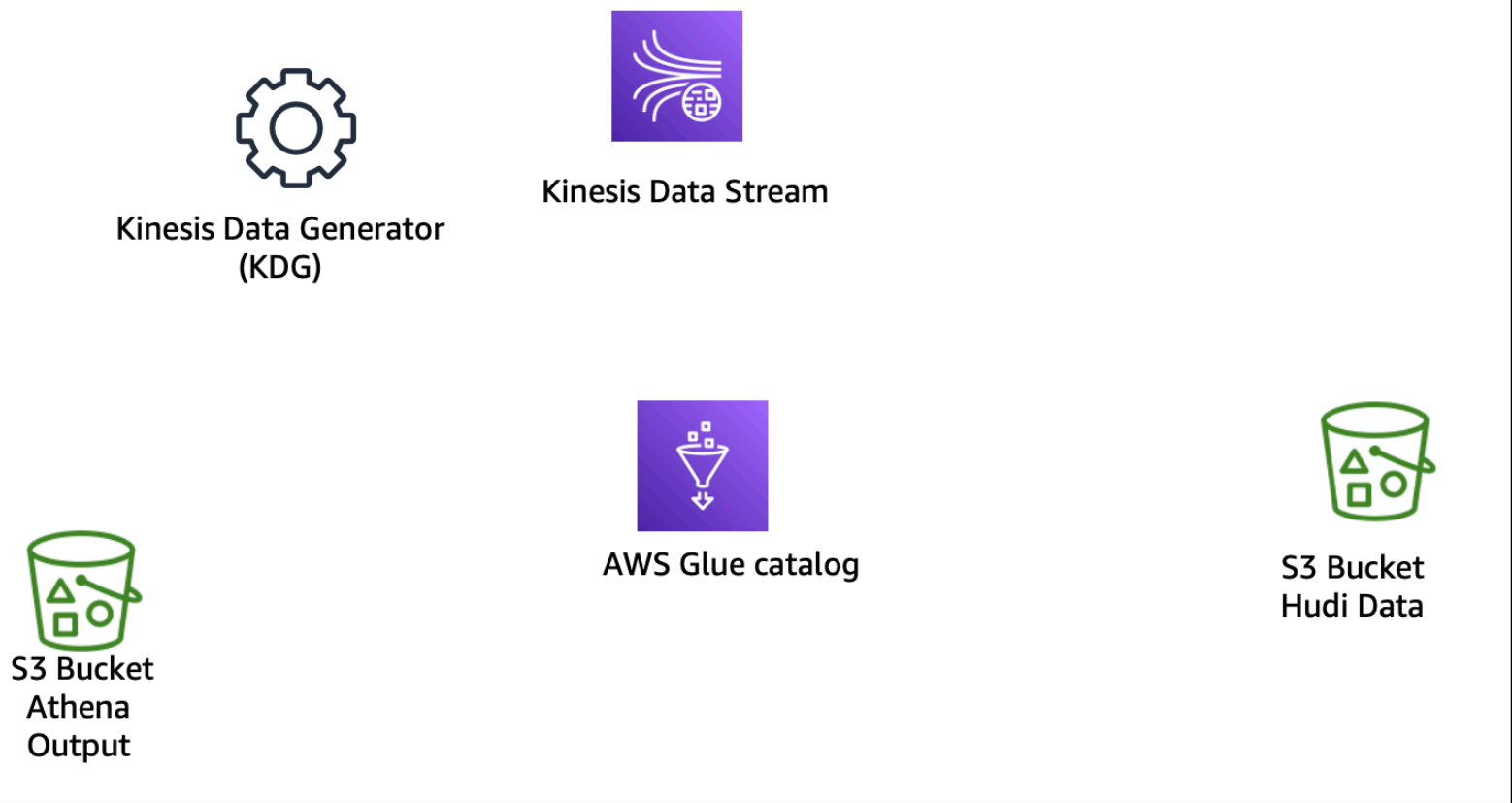
Scenario

Mary is a member of the data science team and works with a lot of streaming data that is collected from IoT devices. Every time the devices are reset, the size and structure of the data changes. A device that normally sends only a few fields on a regular basis might occasionally send several fields. This is complex to handle given that the standard tools expect data to follow a certain structure. Also, the changed data should affect only the requisite rows and not the entire dataset.

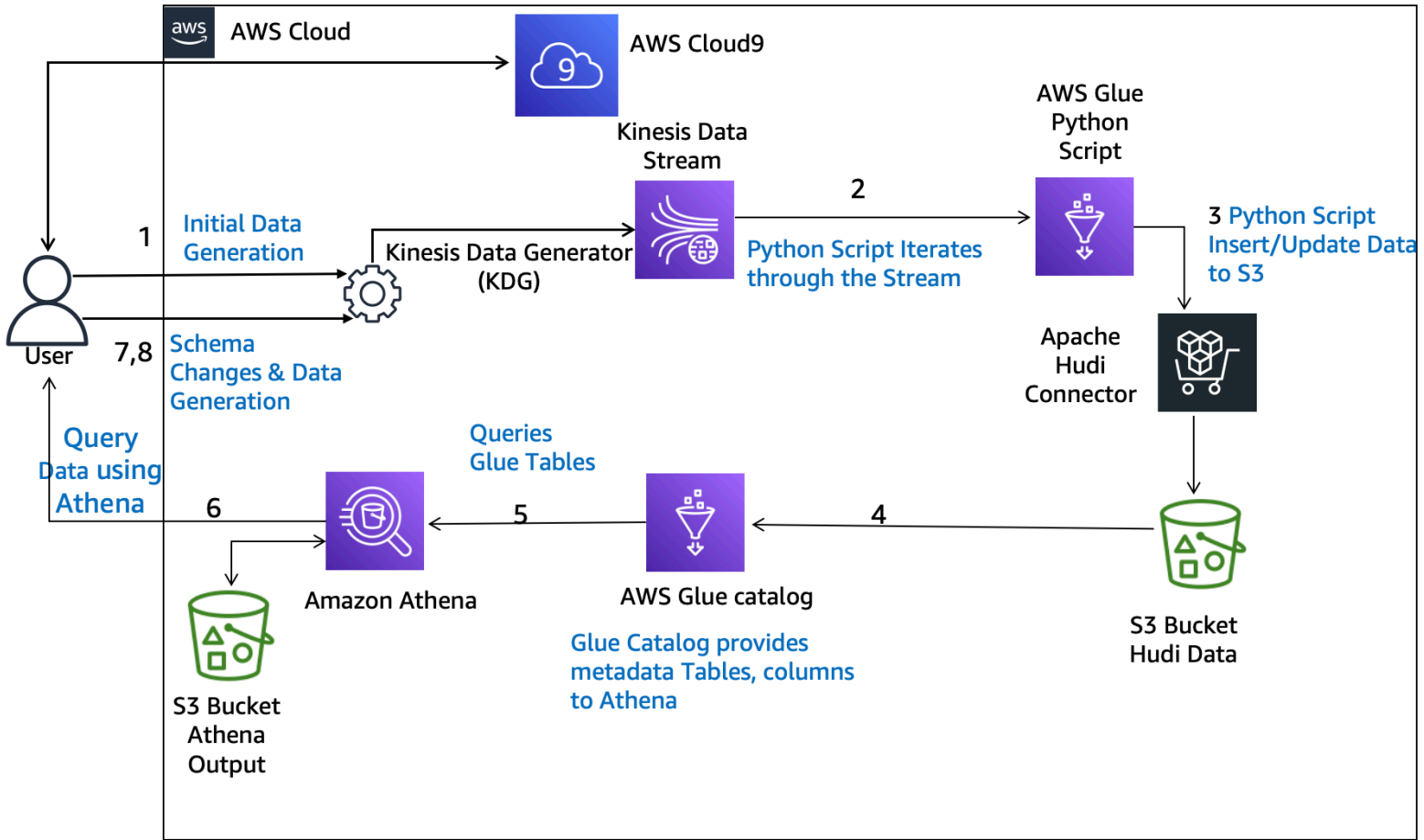
Your challenge is to develop a proof of concept (POC) to accommodate the ever-changing schema and only update the affected records.

You have decided to use an AWS Glue job with custom scripts to handle the dynamic schema and the Apache Hudi Connector for in-place updates for streaming data. You will use Athena to run SQL-like queries on the dynamic data and use Amazon S3 for a data lake. Finally, you will use Amazon Kinesis Data Streams to ingest data that is randomly generated from the Amazon Kinesis Data Generator (KDG), which emulates an IoT device.

When you *start* the lab, the environment will contain the resources that are shown in the following diagram.




By the *end* of the lab, you will have created the architecture that is shown in the following diagram. The table after the diagram provides a detailed explanation of the architecture.




Numbered Step	Detail
1	You start an AWS Glue job. The KDG runs and sends data to a Kinesis data stream.
2	The AWS Glue job runs a Python script to iterate through the stream.
3	A Python script inserts or updates the data in an S3 bucket.
4	The AWS Glue Data Catalog provides metadata, such as tables and column names, to Athena.
5	Athena interacts with Amazon S3 using the metadata that the Data Catalog provided.
6	You run queries in Athena to view the data.
7	You change the schema and run queries to analyze the data.
8	Finally, you revert the schema changes and run queries again to analyze the data.

Accessing the AWS Management Console

1. At the top of these instructions, choose ► **Start Lab**.
 - The lab session starts.
 - A timer displays at the top of the page and shows the time remaining in the session.
💡 Tip: To refresh the session length at any time, choose ► **Start Lab** again before the timer reaches 0:00.
 - Before you continue, wait until the circle icon to the right of the AWS  link in the upper-left corner turns green.
2. To connect to the AWS Management Console, choose the **AWS** link in the upper-left corner.
 - A new browser tab opens and connects you to the console.
💡 Tip: If a new browser tab does not open, a banner or icon is usually at the top of your browser with the message that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and then choose **Allow pop-ups**.

Task 1: Analyzing the lab environment

In this task, you will examine the lab environment and note the details of the initial configuration.

3. Retrieve values for resources that were created in the lab environment.
 - In the search box to the right of  **Services**, search for and choose **CloudFormation** to open the AWS CloudFormation console.
 - In the stacks list, choose the link for the stack name where the **Description** does not contain **ADE**.
 - Choose the **Outputs** tab.
Outputs are listed for some of the resources in the stack as shown in the following image.

Key ▲	Value ▼
AthenaBucketName	ade-dsc-bucket-12c67c10
Cloud9URL	https://us-east-1.console.aws.amazon.com/cloud9/ide/e85ac014616d4b99bbfb10e56e8f580e
DemoKinesisStream	hudi_demo_stream
HUDIBucketName	ade-hudi-bucket-12c67c10
HUDIIamRoleARN	arn:aws:iam::[REDACTED]:role/GlueJobExecutionRole-12c67c10
HudiGluedb	hudi_demo_db
KinesisDataGeneratorUrl	https://awslabs.github.io/amazon-kinesis-data-generator/web/producer.html?upid=us-east-1_iRw3THSzl&ipid=us-east-1:ac766728-17ce-45fe-bb02-782a104bcc19&cid=7sngpkr5jkqjqrpm9apft053j&r=us-east-1

- Copy these values to a text editor to use later in the lab.

The following table briefly describes some of the resources that CloudFormation created for this lab.

Resource	Description
S3 buckets	Used to store data for AWS Glue and Athena
Kinesis data stream	Required to ingest data from the KDG tool
AWS Glue database and table	Required to logically represent the data that is stored in Amazon S3
AWS Glue IAM role	Required to run the AWS Glue job
AWS Cloud9 environment	Required to run commands
Kinesis Data generator	Kinesis Data generator Cognito configuration

In this task, you copied output values from the CloudFormation stack to a text file for later use.

Task 2: Subscribing to and activating the Hudi connector

In this task, you will configure the Hudi connector, which the AWS Glue job will use to interact with data in Amazon S3. With this connection, you can make dynamic in-place data updates. You will configure this tool from the AWS Marketplace.

4. Create the Hudi connector for AWS Glue.

- In the search box at the top, search for and choose **AWS Glue Studio**.
- From the navigation pane choose **AWS Marketplace**.
- In the **Search AWS Glue Studio products** section, search for **hudi**
- In the new window, **Apache Hudi Connector for AWS Glue** is displayed.
- Choose **View purchase options**.
- Choose **Accept Terms**.

Wait for the **Continue to Configuration** button to be available.

- Choose **Continue to Configuration**.
- On the **Launch Apache Hudi Connector for AWS Glue**
 - Under **Setup**, choose **ECS** radio button.
 - For **version**, choose **0.10.1 (Jun 13, 2022)**.
- In the **Launch** section, choose **Activate the Glue connector from AWS Glue Studio**.

The AWS Glue Studio console opens in a new browser tab or window. You will configure and activate the connector here.

- For **Name**, enter `hudi-connection`
- Choose **Create connection and activate connector**.
- To view the details, in the **Connections** section, choose the **hudi-connection** link.

In this task, you provisioned a Hudi connector for AWS Glue to use to interact with an S3 bucket.

Task 3: Configuring job scripts for AWS Glue

In this task, you will retrieve the following files, which are required to configure and run the AWS Glue job.

- [glue_job_script.py](#): This is the Python script that the AWS Glue job will run to perform in-place data updates.
- [glue_job.template](#): This CloudFormation template will be used to create an AWS Glue job.

Note: Choose Right-Click to Save/Open the file and browse the content.

5. Open the AWS Cloud9 terminal and download two files.

- From the text file where you recorded outputs from the CloudFormation stack, find the **Cloud9URL** value. Paste that URL in a new browser tab or window to open the AWS Cloud9 terminal.

Wait for the terminal prompt to display **voclabs:~/environment \$**. It might take a few minutes.

💡 **Tip:** For convenience, you might want to close the *Welcome* tab and drag the terminal window to the top portion of the page.

- To download the files to configure and run the AWS Glue job, run the following commands:

```
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACDENG-1-91570/lab-06-hudi/s3/glue_job_script.py
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACDENG-1-91570/lab-06-hudi/s3/glue_job.template
```

💡 **Tip:** To confirm that both files are successfully downloaded, you can run the `ls` command to list them.

6. Copy the files to an S3 bucket.


- Run the following commands. In both commands, replace `<HUDIBucketName>` with the bucket name that you recorded from the CloudFormation outputs.

```
aws s3 cp glue_job_script.py s3://<HUDIBucketName>/artifacts/
aws s3 cp glue_job.template s3://<HUDIBucketName>/templates/
```

The output from these commands is similar to the following image.

```
voclabs:~/environment $ aws s3 cp glue_job_script.py s3://ade-hudi-bucket-99719360/artifacts/  
upload: ./glue_job_script.py to s3://ade-hudi-bucket-99719360/artifacts/glue_job_script.py  
voclabs:~/environment $ aws s3 cp glue_job.template s3://ade-hudi-bucket-99719360/templates/  
upload: ./glue_job.template to s3://ade-hudi-bucket-99719360/templates/glue_job.template  
voclabs:~/environment $
```

7. Retrieve the URL for the CloudFormation template that you uploaded for the AWS Glue job.

- In the search box to the right of  **Services**, search for and choose **S3** to open the Amazon S3 console.
- Choose the link for the bucket name that contains **ade-hudi-bucket**.
- Choose the **templates** link.
- Select **glue_job.template**, and then choose **Copy URL** to copy the URL for the template. Save the URL to your text editor.

In this task, you configured the scripts that are necessary to create and run the AWS Glue job.

Task 4: Configuring and running the AWS Glue job


AWS Glue provides the capability to run ETL jobs to replicate data across various data sources. The service can run custom and standard ETL scripts. In this task, you will use a CloudFormation template to configure an AWS Glue job. Then, you will run the AWS Glue job, which will run a custom Python script. The script will consume data from the KDG tool and interact with the S3 bucket to insert or update the data as needed.


8. Use Cloudformation to create a stack for the AWS Glue job.

- Navigate to the CloudFormation console.
- Choose **Create stack** > **With new resources (standard)**.
- For **Template source**, choose **Amazon S3 URL**.
- For **Amazon S3 URL**, paste the URL for the CloudFormation template, which you retrieved from Amazon S3 previously.
- Choose **Next**.
- For **Stack name**, enter `create-glue-job`
- For **HudiARN**, paste the **HUDIiamRoleARN** value that you recorded from the CloudFormation outputs.
- For **LocalS3Bucket**, paste the **HUDIBucketName** value that you recorded from the CloudFormation outputs.
- Choose **Next**.
- Choose **Next** again.
- Choose **Create stack**.

Wait for the stack to be created. It might take a few minutes. You might need to refresh the page.

9. Run the AWS Glue job.

- In the search box to the right of  **Services**, search for and choose **AWS Glue** to open the AWS Glue console.
- In the navigation pane, under **ETL**, choose **Jobs**.
- In the **Your jobs** section, choose the link for the job name that contains **Hudi_Streaming_Job**.

 **Note:** The CloudFormation template that you used in the previous step created this job. You can review the Python code on the **Script** tab.

- To start the job, choose **Run** in the upper-right corner.
- To view the status of the job, choose the **Runs** tab.

Before you continue, make sure that the **Run status** is *Running*. You might need to refresh the page.

In this task, you configured and started the AWS Glue job.

Task 5: Using the KDG to send data to Kinesis

In this task, you will use the Kinesis Data Generator (KDG) tool to generate and send random data to Kinesis. The tool will simulate IoT devices sending data from sensors.

10. Access the KDG and start sending data.

- From the outputs that you recorded from CloudFormation, find the **KinesisDataGeneratorUrl** value. Paste that URL in a new browser tab or window to open the KDG tool.

The URL is similar to https://awslabs.github.io/amazon-kinesis-data-generator/web/producer.html?upid=us-east-1_xrN3iZNu2&ipid=us-east-1:dad05a25-1c1a-4efd-b603-4b9e63912446&cid=3090bsfuesh8ui6qdjonu201n6&r=us-east-1.

- Use the following credentials to sign in to the KDG:
 - Username:** Mary
 - Password:** welcome1234
- In the page that displays after you sign in, configure the following:
 - Region:** Choose **us-east-1**.
 - Stream/delivery stream:** Choose **hudi_demo_stream**.
 - Records per second:** Choose **Constant** and enter **1**.
 - For **Record template**, choose **Template 1** and rename it to **Schema 1**
 - Copy and paste the following into the **Schema 1** code block.

```
{
  "name" : "{{random.arrayElement(["Sensor1","Sensor2","Sensor3", "Sensor4"]}}",
  "date": "{{date.utc(YYYY-MM-DD)}}",
  "year": "{{date.utc(YYYY)}}",
  "month": "{{date.utc(MM)}}",
  "day": "{{date.utc(DD)}}",
  "column_to_update_integer": {{random.number(1000000000)}},
  "column_to_update_string": "{{random.arrayElement(["45f","47f","44f", "48f"]}}}"
}
```

- Choose **Send data**.

A window opens and shows that data is being sent to Kinesis. Keep this browser tab or window open to continue sending data to Kinesis as you continue through the lab.

In this task, you configured the KDG and started generating data for Kinesis to consume.

Task 6: Using Athena to inspect the schema and query data

In this task, you will inspect the schema of the table where the data from the KDG is stored. You will also use Athena to run queries on the table.

11. Inspect the table schema.

- Navigate to the AWS Glue console.
- In the navigation pane, choose **Tables**.

Two tables are listed:

- **hudi_demo_table:** This table stores data from the KDG.
- **hudi_demo_kinesis_stream_table:** The AWS Glue job created this table to adapt to NULL values.

Note: It takes few minutes for tables to appear.

- Choose the link for **hudi_demo_table**.

The table schema displays and looks similar to the following image.

Schema Showing: 1 - 12 of 12 < >

	Column name	Data type	Partition key	Comment
1	_hoodie_commit_time	string		
2	_hoodie_commit_seqno	string		
3	_hoodie_record_key	string		
4	_hoodie_partition_path	string		
5	_hoodie_file_name	string		
6	column_to_update_integer	bigint		
7	column_to_update_string	string		
8	date	string		
9	name	string	Partition (0)	
10	year	string	Partition (1)	
11	month	string	Partition (2)	
12	day	string	Partition (3)	

Analysis: Note the four partition columns in the schema. These four columns are used to partition data in the S3 bucket where the data for this table is stored. If you would like, you can examine these partitions in the S3 bucket.

12. Configure Athena.

- In the search box to the right of **Services**, search for and choose **Athena** to open the Athena console.
- In the navigation pane, choose **Query editor**.

💡 **Note:** If the navigation pane is collapsed, choose the menu icon to open it.

- In the **Data** panel, for **Database**, choose **hudi_demo_db**.
- In the **Tables and views** section, expand **hudi_demo_table** to display the schema.

✍️ **Note:** The schema looks like it did when you reviewed it in the AWS Glue console.

- Choose the **Settings** tab at the top of the page.
- Choose **Manage**.
- To the right of the **Location of query result** field, choose **Browse S3**.
- Choose the **ade-dsc-bucket-xxxx**.

✍️ **Note:** Results from Athena queries will be stored in this bucket.

- Select **Choose**, and then choose **Save**.
- Return to the **Editor** tab.
- In the **Data** panel, to the right of **hudi_demo_table**, choose the three dots icon and then choose **Preview Table**.

Notice that the following query appears in the query tab to the right: **SELECT * FROM "hudi_demo_db"."hudi_demo_table" limit 10;**

The query results display and are similar to the following image.

# ▾	_hoodie_commit_time ▾	_hoodie_commit_seqno ▾	_hoodie_record_key ▾	_hoodie_partition_path
1	20220710070942611	20220710070942611_0_8	Sensor4	name=Sensor4/year=2022/month=0
2	20220710071442599	20220710071442599_0_9	Sensor2	name=Sensor2/year=2022/month=0
3	20220710071542616	20220710071542616_0_7	Sensor3	name=Sensor3/year=2022/month=0
4	20220710071142411	20220710071142411_0_8	Sensor1	name=Sensor1/year=2022/month=0

 **Note:** The KDG is simulating IoT devices. In this case, the tool is simulating temperature sensors.

13. Run Athena queries.

- To display the attributes that you are interested in, copy and paste the following query in one of the query tabs, choose 'Run'.

```
SELECT _hoodie_commit_seqno, _hoodie_record_key, column_to_update_string FROM "hudi_demo_table"
```

The results are similar to the following image.

Results (4)				Copy	Download results
<input type="text" value="Search rows"/>				< 1 >	
# ▾	_hoodie_commit_seqno ▾	_hoodie_record_key ▾	column_to_update_string		
1	20220710071742343_0_14	Sensor4	48f		
2	20220710071142411_0_8	Sensor1	44f		
3	20220710071442599_0_9	Sensor2	45f		
4	20220710071542616_0_7	Sensor3	45f		

- Run the query multiple times to see that the values are changing, as shown in the following image.

Results (4)				Copy	Download results
<input type="text" value="Search rows"/>				< 1 >	
# ▾	_hoodie_commit_seqno ▾	_hoodie_record_key ▾	column_to_update_string		
1	20220710072304930_1_10	Sensor2	45f		
2	20220710072304930_0_17	Sensor1	48f		
3	20220710072304930_3_14	Sensor3	44f		
4	20220710072304930_2_17	Sensor4	48f		

 **Note:** Observe the difference in the data that *Sensor 3* is sending.

Analysis: The KDG sends new data each second. Each time you run the query, you get a new random set of four records. The AWS Glue job picks up the data change and uses the Hudi connector to insert or update the data *in place* in the S3 data lake.

In this task, you used Athena to query the data and observed how the data changed.

Task 7: Changing the schema dynamically

In this task, you will change the structure of the data from the KDG. Then, you will run queries in Athena and analyze the results without making any changes to the AWS Glue job or table structure.

14. Change the schema and run Athena queries.

- Return to the tab or window where the KDG tool is running.
- Choose **Stop Sending Data to Kinesis**.
- For **Record template**, choose **Template 2** and rename it to `Schema 2`
- Copy and paste the following into the **Schema 2** code block.

```
{
  "name" : "{{random.arrayElement(["Sensor1","Sensor2","Sensor3", "Sensor4"])}}",
  "date": "{{date.utc(YYYY-MM-DD)}}",
  "year": "{{date.utc(YYYY)}}",
  "month": "{{date.utc(MM)}}",
  "day": "{{date.utc(DD)}}",
  "column_to_update_integer": {{random.number(1000000000)}} ,
  "column_to_update_string": "{{random.arrayElement(["45f","47f","44f","48f"])}}",
  "new_column": "{{random.number(1000000000)}}"
```

 **Note:** An additional column, *new_column*, is included in the schema.

- Choose **Send data**, and keep the KDG tool running.
- Return to the Athena console, and refresh the page.
- In the **Data** panel, in the **Tables and views** section, expand **hudi_demo_table** to display the schema.
Notice that the additional column, *new_column*, is included in the table.
- Run the following query multiple times and observe the *new_column* values changing.

```
SELECT _hoodie_commit_seqno, _hoodie_record_key, column_to_update_string, new_column FROM
"hudi_demo_table"
```

The results are similar to the following.

Results (4)

Copy

Download results

Q Search rows

< 1 >

#	▼	_hoodie_commit_seqno	▼	_hoodie_record_key	▼	column_to_update_string	▼	new_column
1		20220715160607301_0_13		Sensor2		48f		703667673
2		20220715160607301_3_13		Sensor4		45f		385827477
3		20220715160607301_1_11		Sensor3		44f		66365542
4		20220715160607301_2_7		Sensor1		48f		331759100

- Now, run the query again and observe the changes in the *Sensor 3* values.

The results are similar to the following.

# ▾	_hoodie_commit_seqno ▾	_hoodie_record_key ▾	column_to_update_string ▾	new_column ▾
1	20220715160853767_0_18	Sensor3	47f	745907388
2	20220715160853767_2_18	Sensor1	48f	189860282
3	20220715160853767_1_18	Sensor2	44f	941707367
4	20220715160853767_3_20	Sensor4	47f	660716698

Analysis: When the schema was changed to add the new column, the AWS Glue job relied on the schema evolution capabilities that are built in to Hudi. These capabilities enable the update to the AWS Glue Data Catalog to add the new column. Hudi also added the extra column in the output files (Parquet files that are written to Amazon S3). This enables the query engine (Athena) to query the Hudi dataset with an extra column without any issues. For more information, see [Schema Evolution](#) on the Apache Hudi website.

In this task, you modified the schema and observed how the AWS Glue job handled the change. You were able to run Athena queries and perform data analysis without any issues after modifying the schema.

Task 8: Reverting the changes to the schema

In this final task, you will revert the schema and verify that you can continue data analysis without any issues.

15. Change the schema and run Athena queries.

- Return to the tab or window where the KDG tool is running.
- Choose **Stop Sending Data to Kinesis**.
- For **Record template**, choose **Schema 1**.
- Choose **Send data**.
- Choose **Send data**, and keep the KDG tool running.
- Return to the Athena console, and refresh the page.
- In the **Data** panel, in the **Tables and views** section, expand **hudi_demo_table** to display the schema.
Notice that the additional column, *new_column*, is still included in the table.
- Run the following query multiple times.

```
SELECT _hoodie_commit_seqno, _hoodie_record_key, column_to_update_string, new_column FROM  
"hudi_demo_table"
```

Notice that *new_column* is still included in the query results; however, that column doesn't contain any values.

Results (4)

Copy

Download results

Search rows

< 1 > ⚙

# ▾	_hoodie_commit_seqno ▾	_hoodie_record_key ▾	column_to_update_string ▾	new_column ▾
1	20220710073144687_0_27	Sensor1	45f	
2	20220710073144687_3_34	Sensor3	47f	
3	20220710073144687_2_34	Sensor2	47f	
4	20220710073144687_1_22	Sensor4	48f	

Analysis: After you changed the schema again and removed *new_column* from the data, the Python script in the AWS Glue job handled the record layout mismatches.

This method queries the AWS Glue Data Catalog for each to-be-ingested record and gets the current Hudi table schema. It then merges the Hudi table schema with the schema of the to-be-ingested record and enriches that schema with null values for *new_column*. This enables Athena to query the Hudi dataset without any issues.

In this task, you reverted the schema and observed that records were updated in place.

Update from the team

Congratulations! In this lab, you created the Hudi connection and then used a custom Python script to read data from Kinesis Data Streams. You used Athena to run queries on the data and see the changes in near real time. You also changed the schema and observed that Athena queries ran seamlessly and returned the expected data.

Your POC was successful to demonstrate how to process dynamic data changes and accommodate changes to the data structures.

Submitting your work

16. To record your progress, choose **Submit** at the top of these instructions.

17. When prompted, choose **Yes**.

After a couple of minutes, the grades panel appears and shows you how many points you earned for each task. If the results don't display after a couple of minutes, choose **Grades** at the top of these instructions.

💡 **Tip:** You can submit your work multiple times. After you change your work, choose **Submit** again. Your last submission is recorded for this lab.

18. To find detailed feedback about your work, choose **Submission Report**.

Lab complete

Congratulations! You have completed the lab.

19. At the top of this page, choose **End Lab**, and then choose **Yes** to confirm that you want to end the lab.

A message panel indicates that the lab is ending.

20. To close the panel, choose **Close** in the upper-right corner.

© 2022, Amazon Web Services, Inc. and its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.