# Building Streaming Data Analytics Solutions on AWS

# Lab 1: Setting up a Streaming Delivery Pipeline with Amazon Kinesis

© 2024 Amazon Web Services, Inc. or 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. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at *AWS Training and Certification*.

## Lab overview

You are a data platform engineer at AnyCompany Online Shopping. AnyCompany Online Shopping sells a variety of consumer goods through e-commerce transactions in five categories - books, food, apparel, electronics and home goods.

Your company wants to persist the clickstream data, originating from the e-commerce store, in a data store for further analysis to gain actionable insights. You are responsible to collect and persist the clickstream data in Amazon Web Services (AWS) for downstream processing and analysis.

You are exploring the following:

- A fully managed, scalable, and durable real-time data streaming service that can continuously capture streaming data from the producer.
- A fully managed data delivery stream that can transport the streaming data into data lakes.
- A mechanism to dynamically partition the data based on product categories to improve query performance.

You identify *Amazon Kinesis Data Streams* for data ingestion and an *Amazon Kinesis Data Firehose delivery stream* for data delivery purposes.

In this lab, you create and configure a Kinesis Data Firehose delivery stream to connect to a Kinesis data stream. The Kinesis data stream is streaming real-time clickstream data originating from an e-commerce store. You enable and configure dynamic partitioning to create targeted data sets by partitioning the data based on product categories. Partitioning your data minimizes the amount of data scanned, optimizes performance, and reduces costs of your analytics queries on Amazon S3. Finally, you configure destination settings for the delivery stream to transfer the output to Amazon Simple Storage Service (Amazon S3).

### OBJECTIVES

By the end of this lab, you will be able to do the following:

- Create a Kinesis Data Firehose stream and connect the Kinesis data stream to Kinesis Data Firehose.
- Configure dynamic partitioning on the Kinesis Data Firehose delivery stream.
- Deliver data to Amazon S3.

## ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

- **Expected output:** A sample output that you can use to verify the output of a command or edited file.
- **Note:** A hint, tip, or important guidance.
- **Warning:** Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).
- **Task complete:** A conclusion or summary point in the lab.

# Start lab

1. To launch the lab, at the top of the page, choose Start lab.

**Caution:** You must wait for the provisioned AWS services to be ready before you can continue.

2. To open the lab, choose Open Console.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**WARNING: Do not change the Region unless instructed.**

## COMMON SIGN-IN ERRORS

### Error: You must first sign out


If you see the message, **You must first log out before logging into a different AWS account:**

- Choose the **click here** link.
- Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
- Choose Open Console again.

### Error: Choosing Start Lab has no effect

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

- Add the lab domain name to your pop-up or script blocker's allow list or turn it off.
- Refresh the page and try again.

## LAB ENVIRONMENT

The following diagram shows the basic architecture of the lab environment.

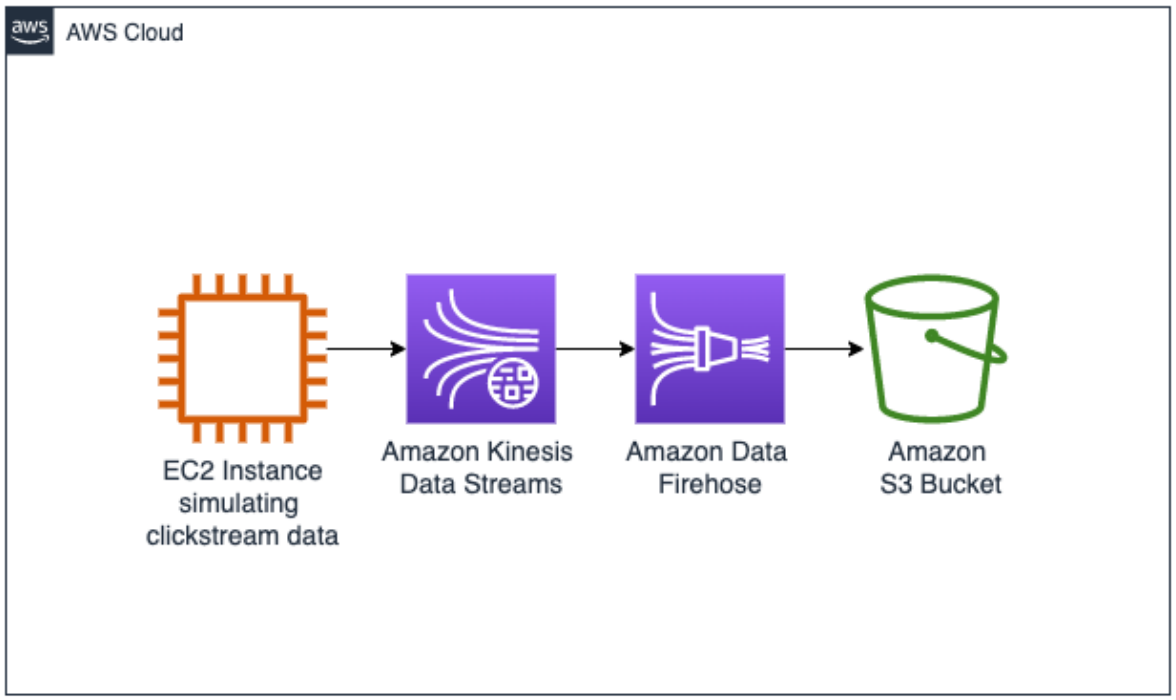



*Image: the following list details the major resources in the diagram:*

- An *EC2 instance* that functions as a clickstream generator. This has been pre-provisioned in this lab.
- An *Amazon Kinesis Data Stream* that collects the streaming data from the EC2 instance. This has been pre-provisioned in this lab.
- An *Amazon Data Firehose Delivery Stream* that delivers data to Amazon S3 bucket.
  - **Note:** You setup and configure the delivery stream in this lab.
- An *Amazon S3 bucket* where the data persists. This has been pre-provisioned in this lab.

## AWS SERVICES NOT USED IN THIS LAB

AWS service capabilities used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

# Task 1: Simulate clickstream data generation

In this task, you connect to a custom producer running on an EC2 instance using Session Manager, a capability of AWS Systems Manager, and follow the steps to start the clickstream generator.

Imagine owning a website or a web app. Users find their way to your site and perform various actions. Visiting a page, clicking a button, submitting information, and purchasing a product are examples of such activities. The users could also be using different devices in performing these operations. The simulator tries to imitate this payload structure across five sample categories (food, electronics, apparel, books, and home goods).

**Note:**

- Session Manager provides secure instance management without needing to open inbound ports, maintain bastion hosts, or manage Secure Shell (SSH) keys.
- If the command shell is unresponsive when you type a command, to return to the shell prompt, press **Ctrl+C**.

3. To access **Session Manager**, in the list to the left of these instructions, copy the **CommandHostSessionManagementURL**.
4. To display the **EC2 Producer terminal** window, paste the **URL** into a new browser tab and press **Enter**.

The Kinesis stream name is needed for the clickstream data ingestion. This value is dynamically obtained and set as a variable in the commands below. It displays the name so you can ensure the value is set. Then it initiates the script that ingests the data.

5. **Command:** To start the **clickstream_generator_items.py** script, run the following command:

```
STREAM_NAME=$(aws kinesis list-streams --query "StreamNames[?contains(@,
'KdsClickstreamData')]" --output text)

echo -e "\n\nThe stream name is : $STREAM_NAME\n\n"

python3 clickstream_generator_items.py $STREAM_NAME 1 1
```

**Expected output:** Output has been shortened as it just keeps printing.

```
******************************
**** This is OUTPUT ONLY. ****
******************************


The stream name is : LabStack-jfoekb-3o57AGNcJdbHE3CcGfwQ6J-0-
KdsClickstreamData-8VgwXqv9PuRr


 Number of arguments: 4 arguments

 Argument List: ['clickstream_generator_items.py', 'LabStack-jfoekb-
3o57AGNcJdbHE3CcGfwQ6J-0-KdsClickstreamData-8VgwXqv9PuRr', '1', '1']

 The program name is: clickstream_generator_items.py
```

```
 The kinesis data stream name is : LabStack-jfoekb-3o57AGNcJdbHE3CcGfwQ6J-0-
KdsClickstreamData-8VgwXqv9PuRr

 Max interval in seconds between records : 1
The current region is set to us-west-2.

{"event_id": "6136066c482ddd74a268451d43454187", "event": "reviewed_item",
"user_id": 1, "item_id": 31, "item_quantity": 0, "event_time": "2023-09-27
16:35:26.025579", "os": "ios", "page": "apparel", "url": "www.example.com"}
{"event_id": "8200e635814059ec209e1ca4d41d36e8", "event": "liked_item",
"user_id": 30, "item_id": 13, "item_quantity": 0, "event_time": "2023-09-27
16:35:27.143534", "os": "android", "page": "home", "url": "www.example.com"}
{"event_id": "444837c2003ade27d0fc3fba140901bb", "event": "reviewed_item",
"user_id": 2, "item_id": 52, "item_quantity": 0, "event_time": "2023-09-27
16:35:28.154368", "os": "ios", "page": "electronics", "url":"www.example.com"}
{"event_id": "d027166efbd8107f9c87e35ec6896276", "event": "reviewed_item",
"user_id": 20, "item_id": 13, "item_quantity": 0, "event_time": "2023-09-26
17:38:27.620738", "os": "web", "page": "books", "url": "www.example.com"}
{"event_id": "9e05835423cd2f2ce11794890a16d006", "event": "purchased_item",
"user_id": 36, "item_id": 33, "item_quantity": 4, "event_time": "2023-09-26
17:38:27.631147", "os": "android", "page": "books", "url":"www.example.com"}
{"event_id": "b5e17e03743ac37b04e697e6ab6b6796", "event":
"clicked_item_description", "user_id": 48, "item_id": 23, "item_quantity": 0,
"event_time": "2023-09-26 17:38:28.642577", "os": "web", "page": "books",
"url": "www.example.com"}
{"event_id": "0faf5f8a5931d80b6ff46c9486c4d230", "event":
"clicked_item_description", "user_id": 44, "item_id": 33, "item_quantity": 0,
"event_time": "2023-09-26 17:38:29.652591", "os": "android", "page": "food",
"url": "www.example.com"}
{"event_id": "0433fe4d36d01c3b94523119b632c68f", "event":
"clicked_item_description", "user_id": 34, "item_id": 23, "item_quantity": 0,
"event_time": "2023-09-26 17:38:30.662130", "os": "android", "page": "food",
"url": "www.example.com"}
```

**Optional:** The response from Kinesis Data Stream can be observed by appending `--verbose` to the end of the command.

**Expected output:** This is output for one shardId.

```
**************************************************
**** This is the Kinesis response OUTPUT ONLY. ****
**************************************************

{
 "event_id": "9a75230c6e99eb57c6d5a39595474291",
 "event": "clicked_review",
 "user_id": 25,
 "item_id": 12,
 "item_quantity": 0,
 "event_time": "2023-09-26 17:46:03.970616",
 "os": "web",
 "page": "electronics",
 "url": "www.example.com"
}
Message written to the Kinesis Data Stream
```

```
{
 "ShardId": "shardId-000000000001",
 "SequenceNumber": "49644893285493347387758872903216807650192741278821646354",
 "EncryptionType": "KMS",
 "ResponseMetadata": {
 "RequestId": "f918f3c1-3253-7cfa-a495-e8da46c1bf61",
 "HTTPStatusCode": 200,
 "HTTPHeaders": {
  "x-amzn-requestid": "f918f3c1-3253-7cfa-a495-e8da46c1bf61",
  "x-amz-id-2":
"buizxhEv/aaPHfVa9SuG5sHE4V2Xbl3YDh8/FyIJhzpZpl+eeK/39vOjeCOqLkg6wapDcA+lTjkBd
YpLbd5VVv9KhXJZDA9/",
  "date": "Tue, 26 Sep 2023 17:46:03 GMT",
  "content-type": "application/x-amz-json-1.1",
  "content-length": "133"
 },
 "RetryAttempts": 0
 }
}
shardId-000000000001
SequenceNumber:
49644893285493347387758872903216807650192741278821646354
HTTStatusCode:
200
```


**Note:**

- Keep the web browser tab open for the duration of this lab.
- If the Session Manager window times out, repeat this task to start producing the clickstream data again.

**Task complete:** You have successfully simulated clickstream data generation by running the clickstream generator Python script on the EC2 Producer terminal.

# Task 2: Create and configure a Kinesis Data Firehose delivery stream

In this task, you create a Kinesis Data Firehose delivery stream and connect the source Kinesis data stream to the delivery stream. With Kinesis Data Firehose, there is no need to write applications or manage resources. You configure your data producers to send data to Kinesis Data Firehose, and it automatically delivers the data to the destination that you specified.

## TASK 2.1: CREATE A DELIVERY STREAM

6. Navigate back to the **AWS Management Console** tab.
7. At the top of the Console, in the search bar, search for and choose

   Amazon Data Firehose. Open the link in a new tab.
8. Choose **Create Firehose stream**.

The browser takes you to the **Create Firehose stream** page where you choose options under the **Choose source and destination** section.

9. For **Source**, choose **Amazon Kinesis Data Streams**.
10. For **Destination**, choose **Amazon S3**.
11. In the **Source settings** section, make the following selections:

- For **Kinesis data stream**, choose **Browse**.
- Choose the radio button next to the only kinesis data stream that has **KdsClickstreamData** in its name.
- Select **Choose**.

12. For the **Firehose stream name**, enter

FH-Stream-Kinesis.

You now specify an S3 bucket as the destination for the Kinesis Data Firehose delivery stream.

13. In the **Destination settings** section:

For **S3 bucket**, choose **Browse**.

14. Choose the radio button next to the S3 Bucket with the name that starts with **databucket-**.
15. Select **Choose**.
16. To enable dynamic partitioning on the incoming data stream by **page** and **event**, in the **Dynamic partitioning** section, choose the radio button next to **Enabled**.

Partitioning by *page* and *event* makes for efficient query optimization in this particular use case.

17. Enable **Inline parsing for JSON** by selecting the radio button next to **Enabled**.

**Note:** This method uses the Kinesis Data Firehose built-in support mechanism, a jq parser, for extracting the keys for partitioning data records that are in JSON format.

18. Specify the **key name** and **JQ expression** as below:

- For **Key name**, enter

page.

- For JQ expression, enter

.page

- Choose **Add dynamic partitioning key**.
- For **Key name**, enter

event. For JQ expression, enter

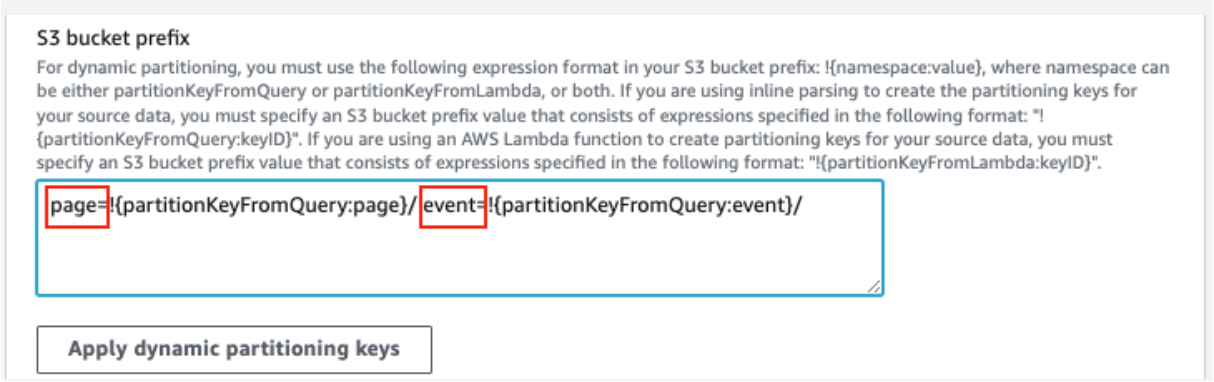
.event.

19. In the **S3 bucket prefix** section, choose **Apply dynamic partitioning keys** .
20. To enable Hive compatible style partitioning by *page* and *event*, update the **S3 bucket prefix** default value with

    `page=` and

    `event=` by pasting the following snippet:

```
page=!{partitionKeyFromQuery:page}/event=!{partitionKeyFromQuery:event}/
```




21. In the **S3 bucket error output prefix** box, enter

    `kdferror/` .

**Note:** The S3 error bucket prefix contains all the records that Kinesis Data Firehose is not able to deliver to the specified S3 destination.

You can now configure options for the delivery stream.

22. Expand the **Buffer hints, compression and encryption** section.
23. For **Buffer size**, enter

    `64` .
24. For **Buffer interval**, enter

    `60` .

**Note:** Kinesis Data Firehose buffers incoming streaming data to a certain size and for a certain period of time before delivering it to the specified destinations. For a delivery stream where data partitioning is enabled, the buffer size ranges from 64 to 128MB, with the default set to 128MB, and the buffer interval ranges from 60 seconds to 900 seconds. Given the time constraints of this lab, you set the buffer size and buffer interval to the minimum values allowed.

You can now set up access permissions.

25. Expand **Advanced Settings**.
26. In the **Service access** section, select the radio button next to **Choose existing IAM role**.
27. For **Existing IAM roles**, choose the **FirehoseRole**.

**Note:** This is the role that Kinesis Data Firehose uses to access your S3 bucket. This has been pre-provisioned for use in this lab.

28. Review the stream configuration, and then choose **Create Firehose stream**.

**Note:** It can take up to 5 minutes for the Firehose stream to be provisioned. Notice the status change from *Starting* to *Active* in the **Firehose stream details** section of the page.

**Task complete:** You created a Kinesis Data Firehose delivery stream and connected the source Kinesis data stream to the delivery stream. You configured your data producers to send data to Kinesis Data Firehose to automatically deliver the data to the destination that you specified.

# Task 3: Verify your output in Amazon S3

In this task, you verify that the output in Amazon S3 is partitioned into *page* and further into *events* in accordance with the configured dynamic partitioning.

## TASK 3.1: LOCATE OUTPUT IN AMAZON S3

29. Navigate back to the AWS Management Console tab.
30. At the top of the console, in the search bar, search for and choose

    S3

    .
31. Open the link in a new tab.
32. Choose the text link for S3 bucket starting with **databucket-** to examine its content.

**Note:** Choose the refresh icon until you see the folders with hive style partitioning. It can take up to 5 minutes for the output to be available in the S3 bucket.

33. Notice that the incoming data stream has been dynamically partitioned into **page/** and within the page into **event/**.

## TASK 3.2: QUERY DATA WITH S3 SELECT

34. Navigate through the folders in the S3 bucket. Choose any of the categories and then choose an event.
35. Select the box next to any object in the folder.
36. In the **Actions** menu, choose **Query with S3 Select**.
37. In the **Input settings**, for **Format**, choose **JSON**.
38. In the **Output settings**, for **Format**, choose **JSON**.
39. In **SQL query**, choose **Run SQL query** . This returns a JSON output limited to five results.
40. The **Query results** returns a JSON output of the event_id, event, user_id, event_time, os, page, and url.

**Expected output:**

```
******************************
**** This is OUTPUT ONLY. ****
******************************
```

```
{
 "event_id": "3a27417ad8a26ad800f9c5a376623183",
 "event": "clicked_item_description",
 "user_id": 28,
 "item_id": 43,
 "item_quantity": 0,
 "event_time": "2023-09-25 21:59:46.379948",
 "os": "web",
 "page": "apparel",
 "url": "www.example.com"
}
{
 "event_id": "220fae4f6cafa03f7b70fa6214bf5357",
 "event": "clicked_item_description",
 "user_id": 21,
 "item_id": 11,
 "item_quantity": 0,
 "event_time": "2023-09-25 21:59:49.457759",
 "os": "web",
 "page": "apparel",
 "url": "www.example.com"
}
{
 "event_id": "6488f278f9b9ddf9ce8eea5d36bbecba",
 "event": "clicked_item_description",
 "user_id": 2,
 "item_id": 42,
 "item_quantity": 0,
 "event_time": "2023-09-25 22:00:10.923585",
 "os": "ios",
 "page": "apparel",
 "url": "www.example.com"
}
{
 "event_id": "0f4887274db86e75ae0cc83ef8b8127f",
 "event": "clicked_item_description",
 "user_id": 24,
 "item_id": 23,
 "item_quantity": 0,
 "event_time": "2023-09-25 22:01:00.845153",
 "os": "android",
 "page": "apparel",
 "url": "www.example.com"
}
```


**Task complete:** You have successfully built a streaming pipeline with Kinesis Data Firehose to connect to the Kinesis data stream and deliver the data to Amazon S3. You have also configured Kinesis Data Firehose to dynamically partition data based on product category to improve query performance, thereby meeting all the requirements set forth by your company.

# Conclusion

You have done the following:

- Created a Kinesis Data Firehose stream and connected the Kinesis data stream to Kinesis Data Firehose.
- Configured dynamic partitioning on the Kinesis Data Firehose delivery stream.
- Delivered data to Amazon S3.

# End lab

Follow these steps to close the console and end your lab.

41. Return to the **AWS Management Console**.
42. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
43. Choose End lab and then confirm that you want to end your lab.