DeepLearning.AI

Data Ingestion

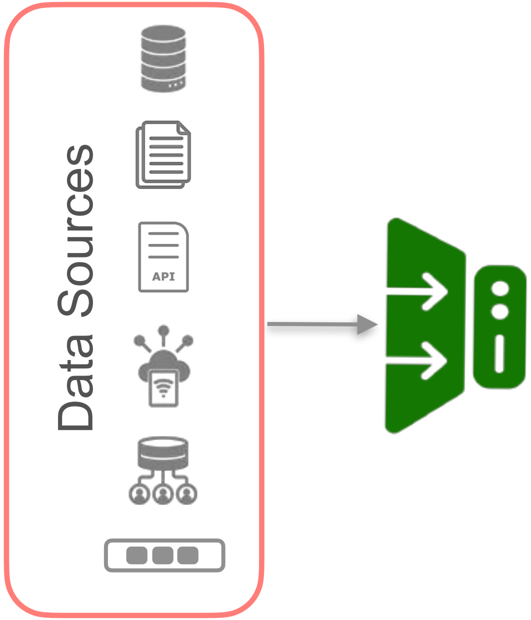
Week 2

DeepLearning.AI

Data Ingestion Overview

Week 2 Overview

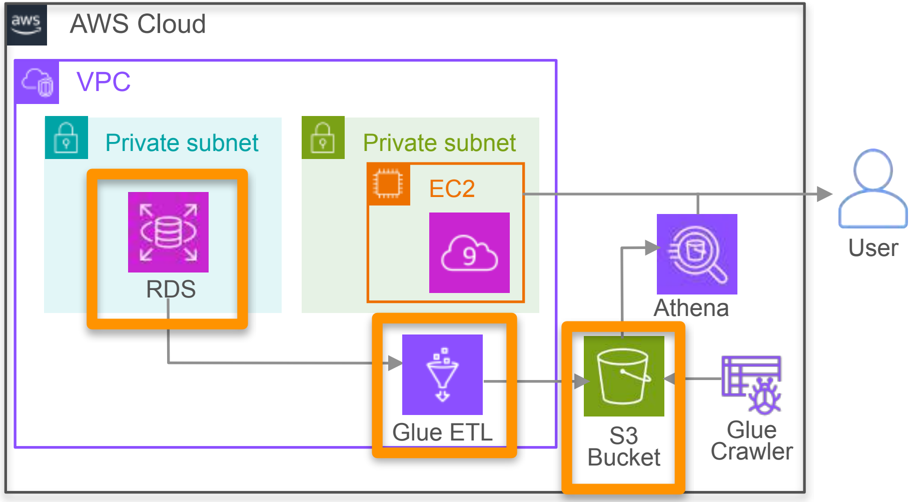
Data Ingestion



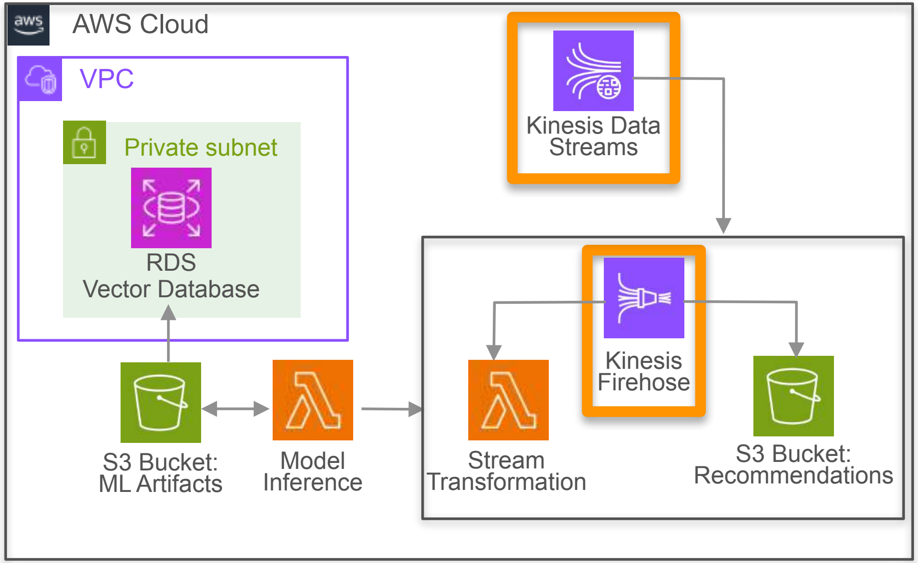
Get raw data → Turn it into something useful → Make it available for downstream use cases

Data Ingestion

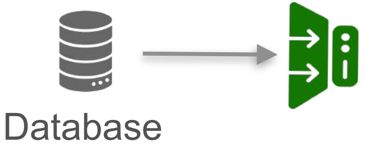
Course 1 Week 2 Lab



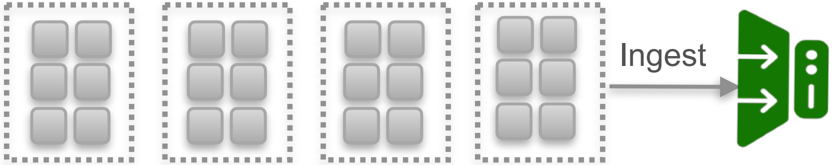
Course 1 Week 4 Lab



Course 2 Week 1 Lab

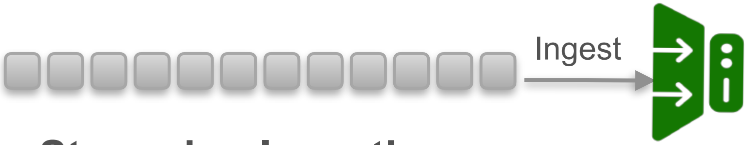


Week 2 Overview



Batch Ingestion

Processing data in chunks or batches



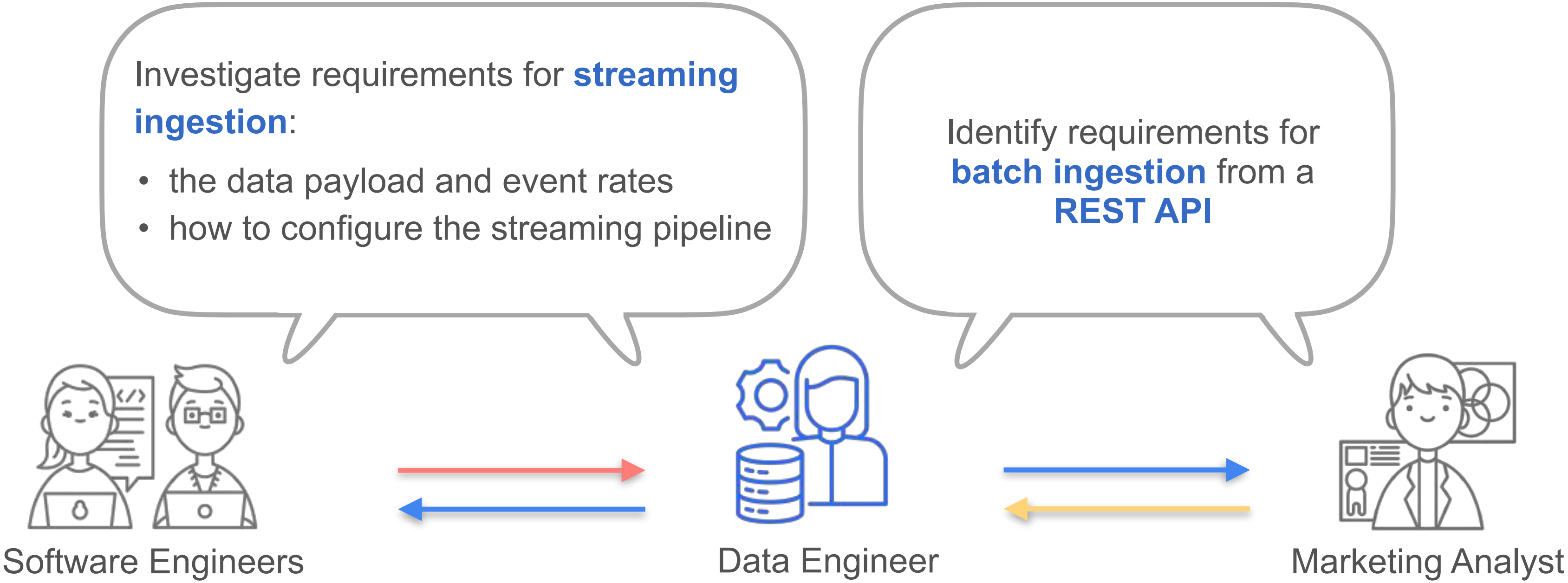
Streaming Ingestion

Processing a continuous stream of events

Week 2 Overview

Lab: streaming ingestion

Lab: batch ingestion



Investigate requirements for **streaming ingestion**:

- the data payload and event rates
- how to configure the streaming pipeline

Identify requirements for **batch ingestion** from a **REST API**

Software Engineers

Data Engineer

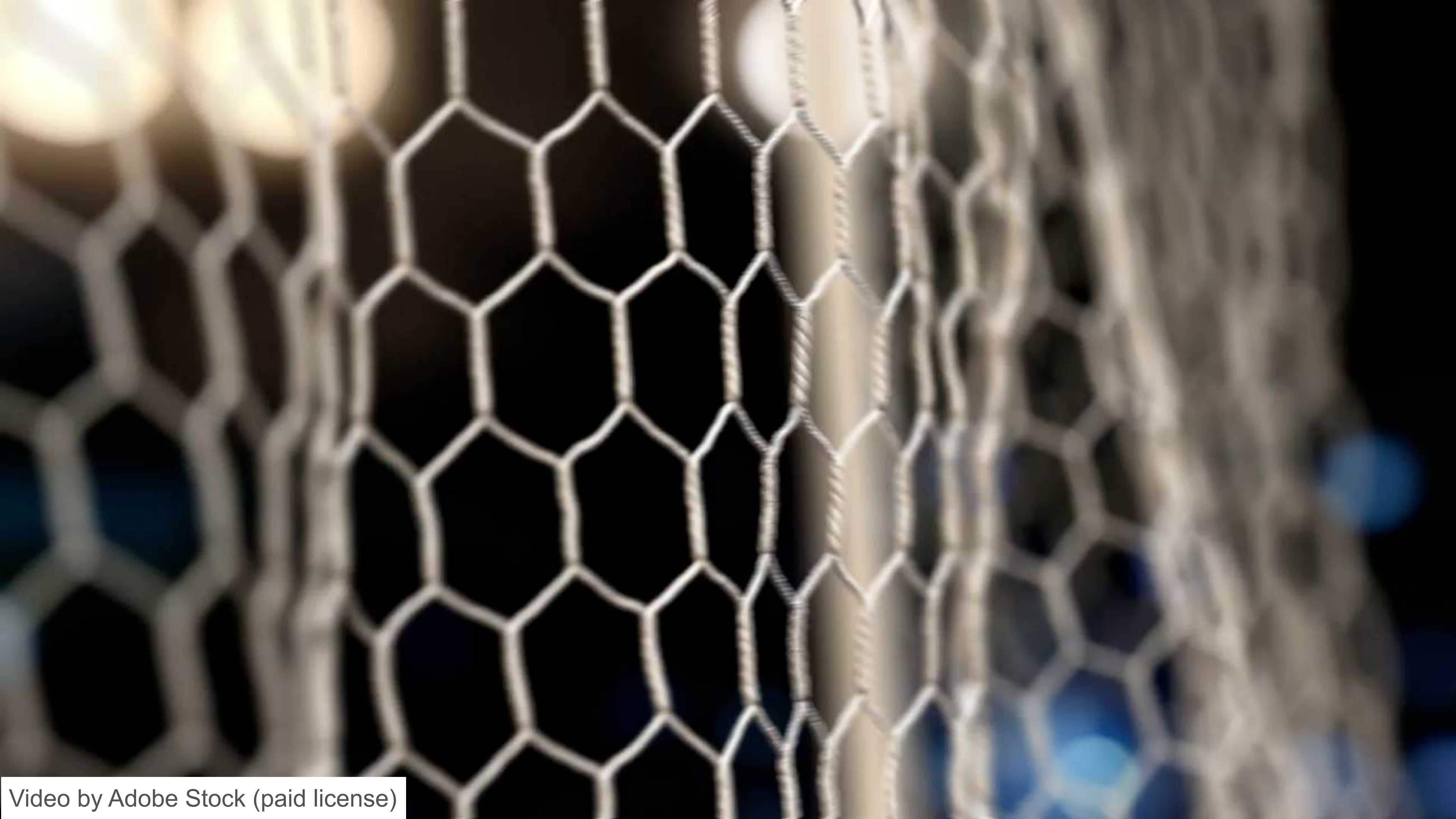
Marketing Analyst

DeepLearning.AI

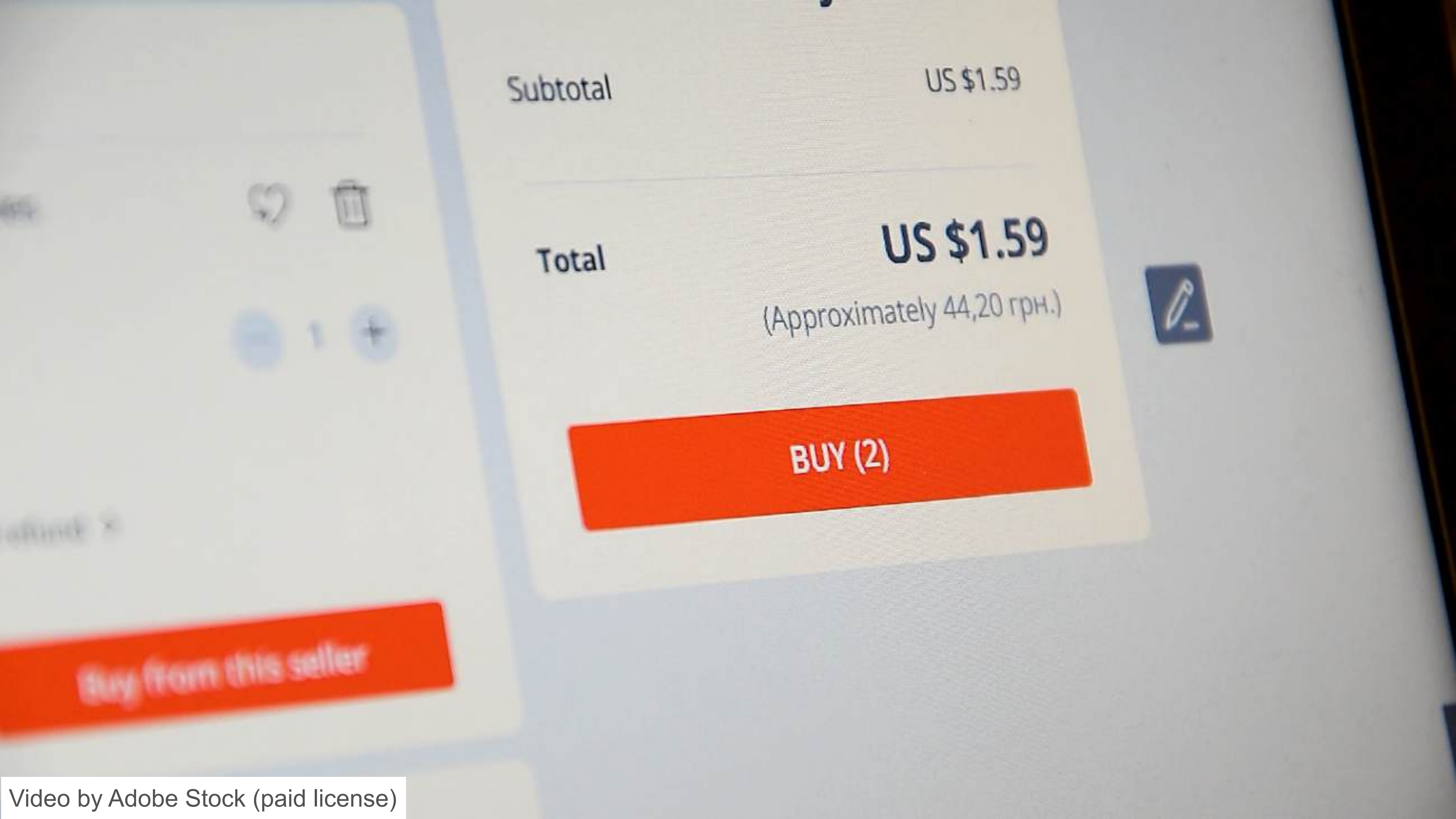
Data Ingestion

Data Ingestion on a Continuum

Video by Adobe Stock (paid license)



Video by Adobe Stock (paid license)



Subtotal US $1.59

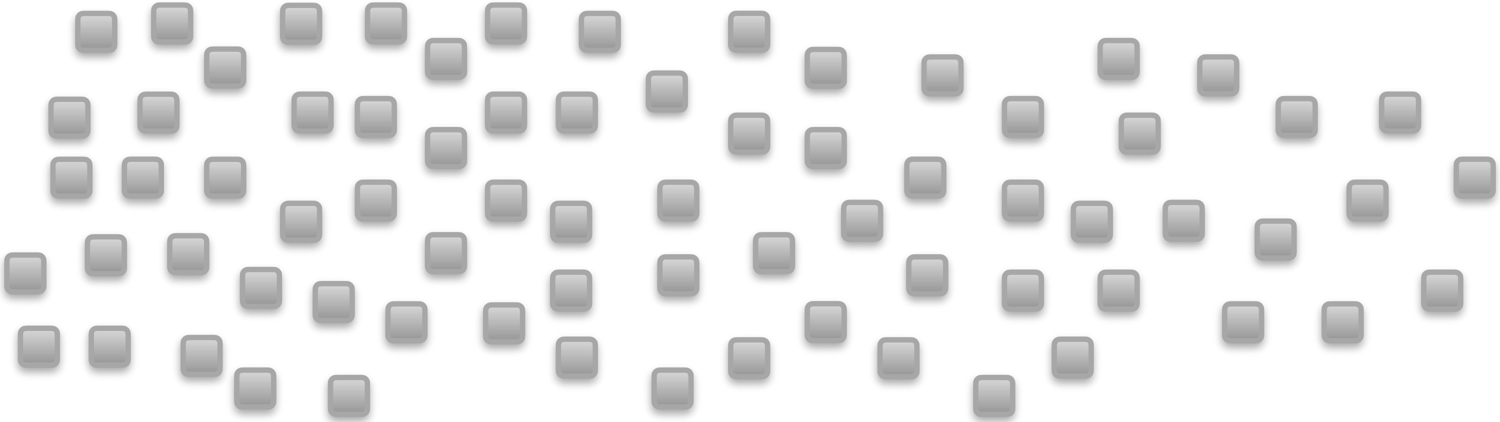
Total US $1.59

(Approximately 44,20 грн.)

Video by Adobe Stock (paid license)

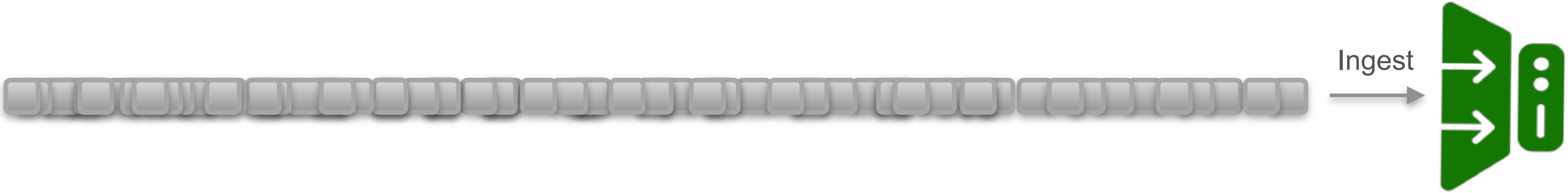
Data Ingestion

Unbounded Data: continuous stream of events

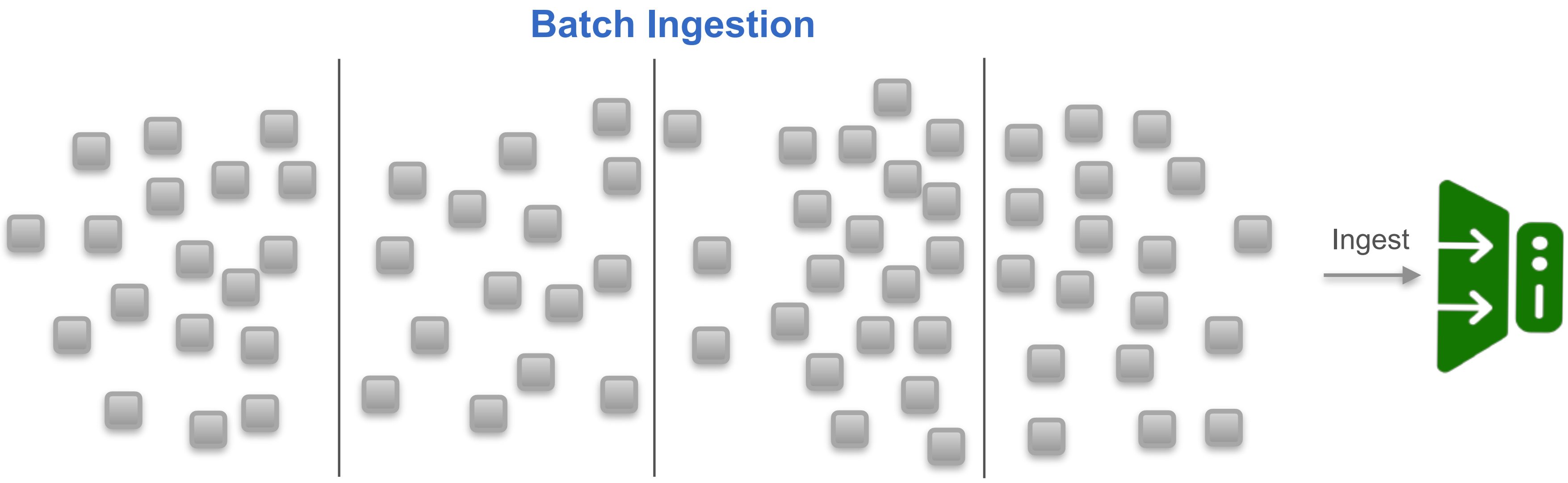


Data Ingestion

Stream Ingestion

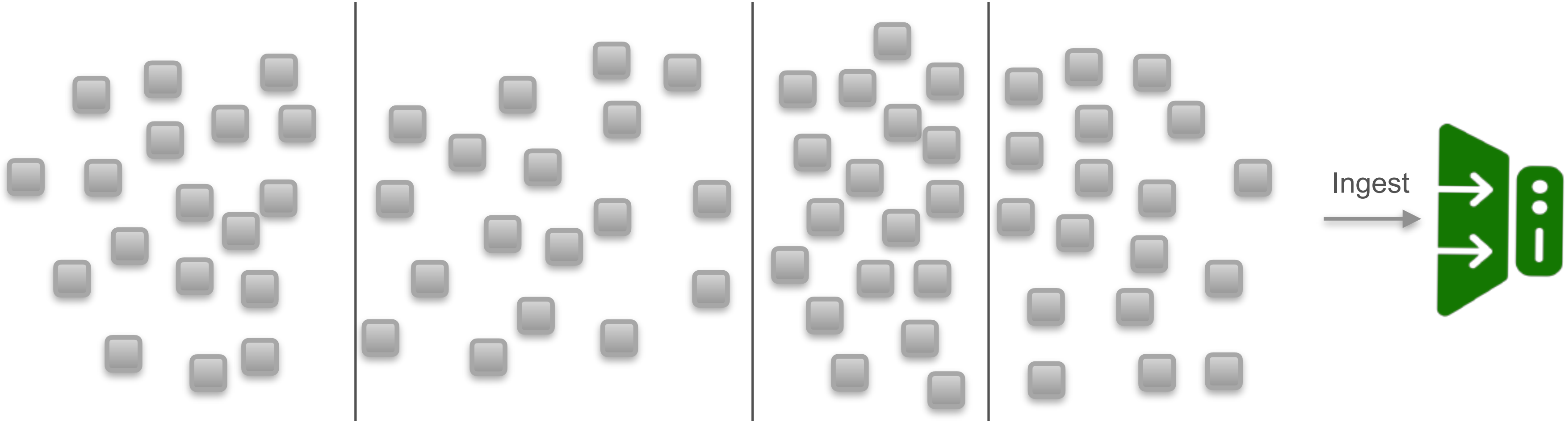


Data Ingestion



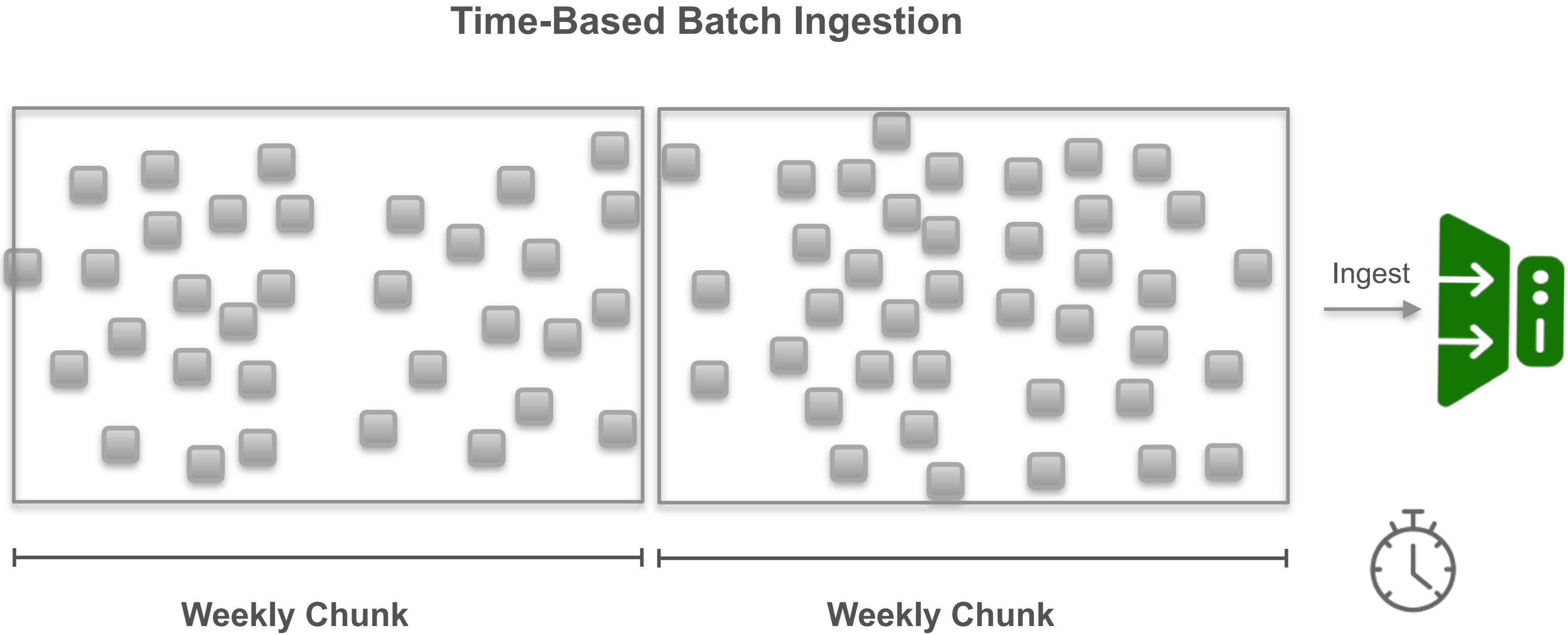
Data Ingestion

Size-Based Batch Ingestion



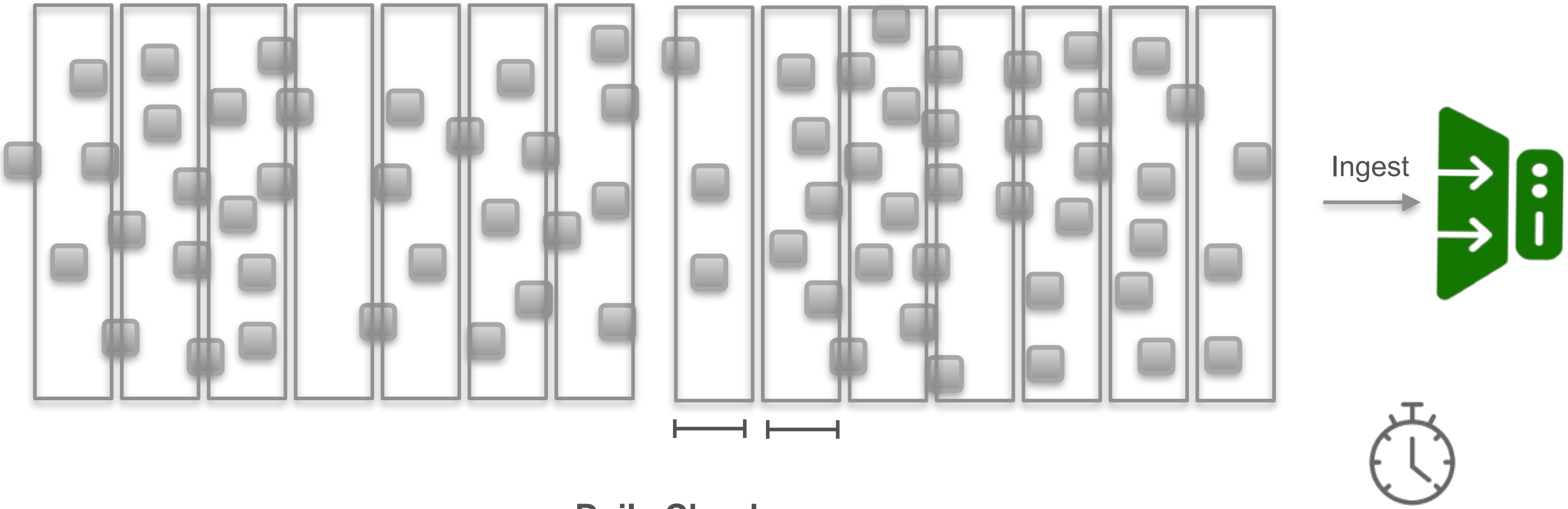
10 GB	10 GB	10 GB	10 GB	**Size-threshold**
1000 events	1000 events	1000 events	1000 events	**Total number of records**

Data Ingestion



Data Ingestion

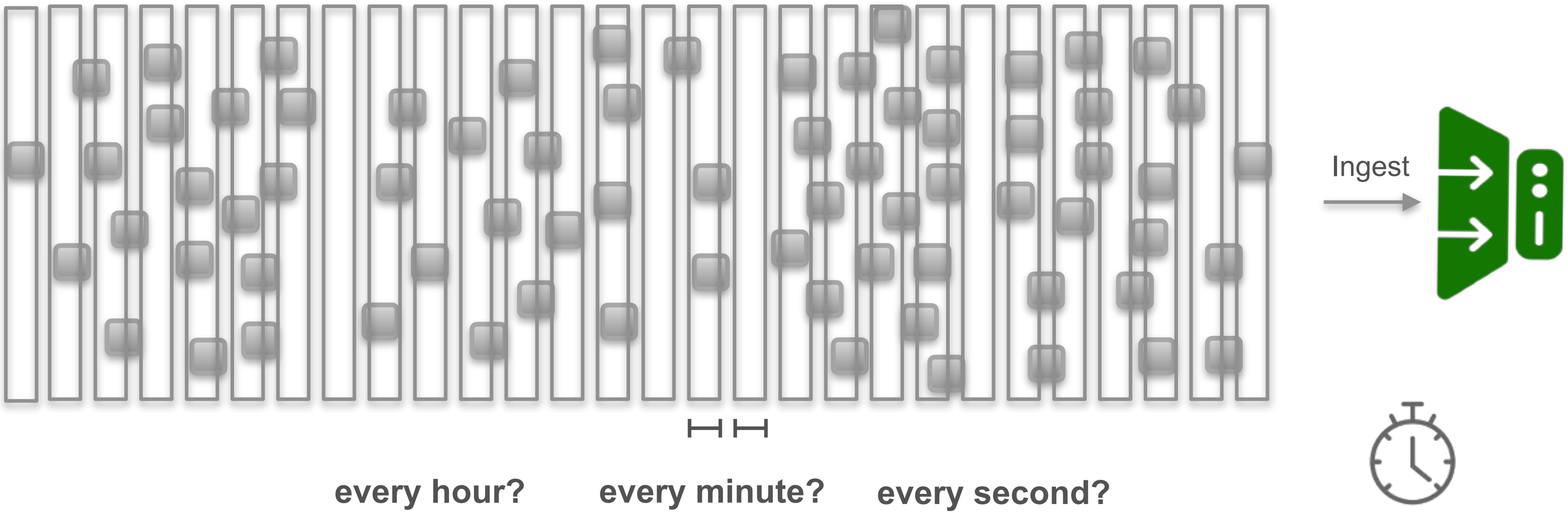
Time-Based Batch Ingestion



Daily Chunks

Data Ingestion

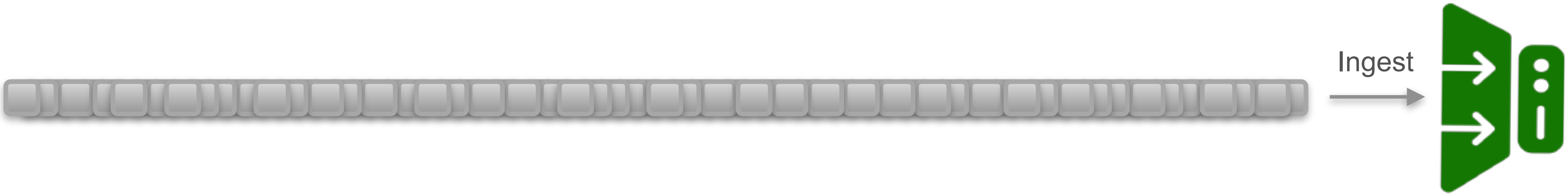
Time-Based Batch Ingestion



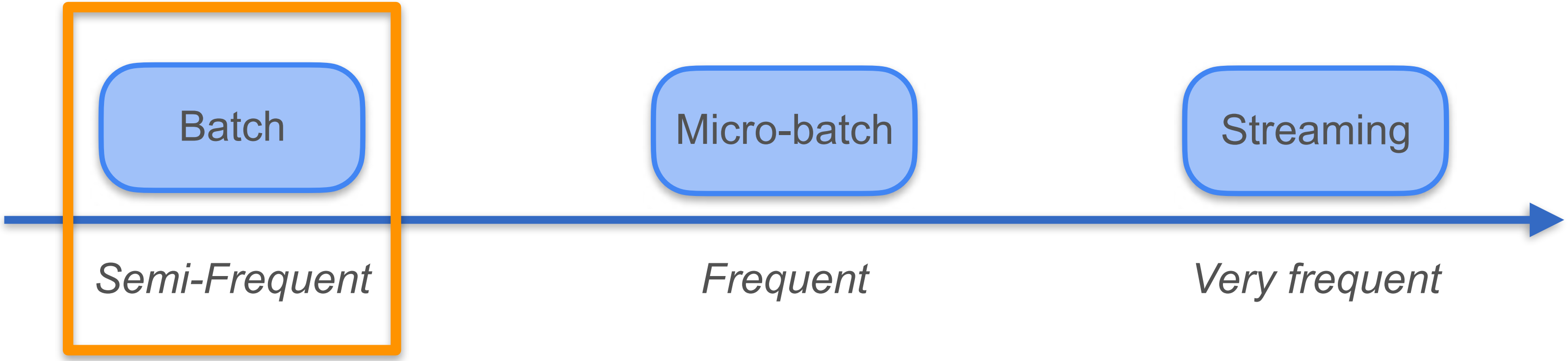
every hour? every minute? every second?

Data Ingestion

Stream Ingestion



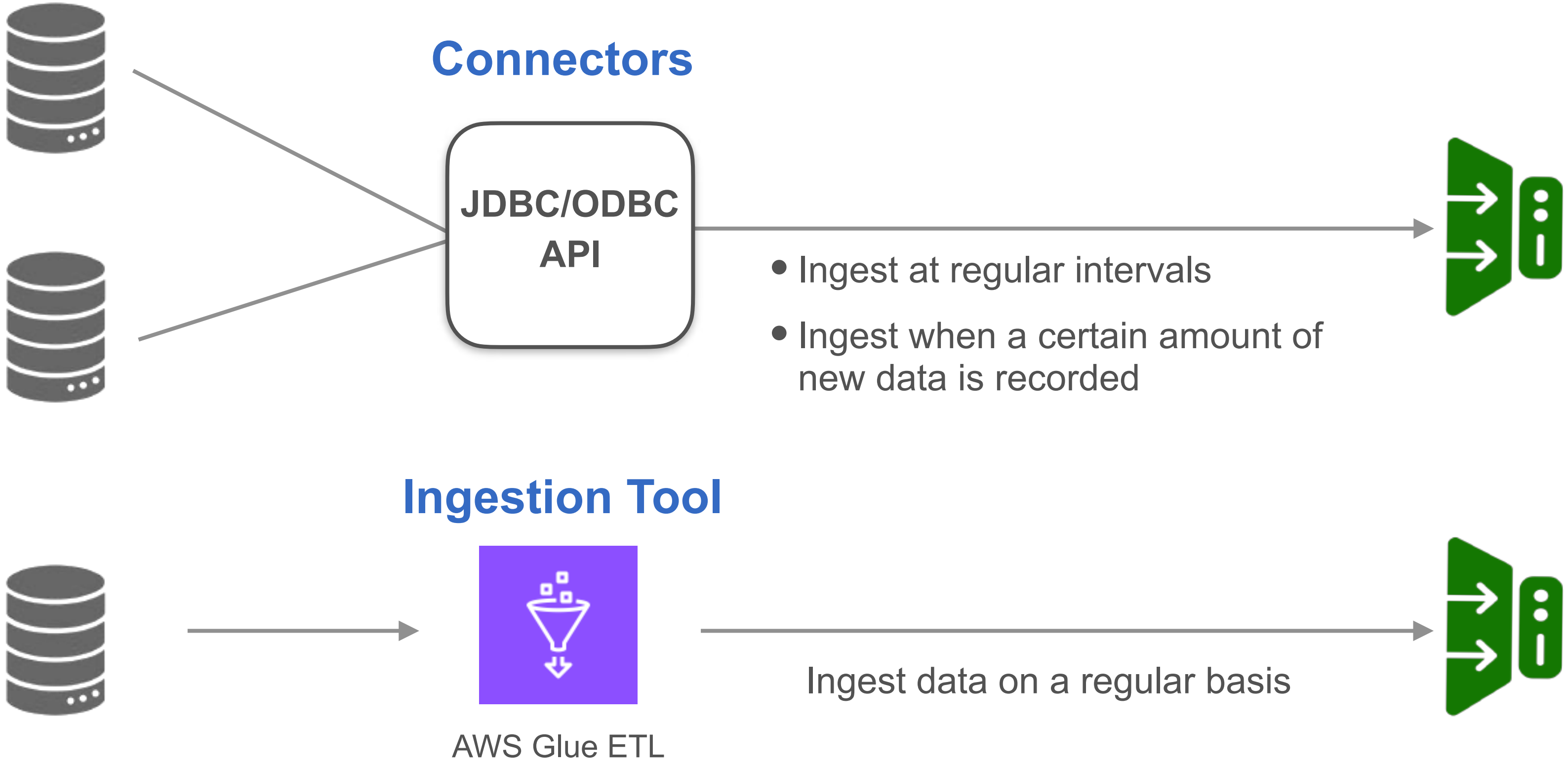
Ingestion Frequencies



Choice of ingestion frequency depends on:

- the source systems you’re working with
- the end use case

Ways to Ingest Data from Databases



Connectors

JDBC/ODBC API

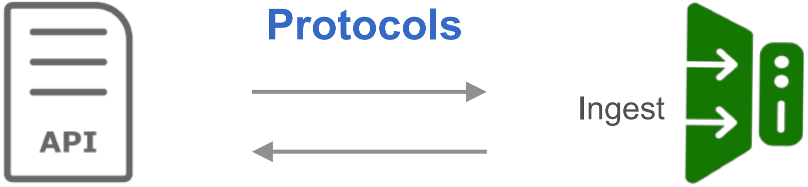
- Ingest at regular intervals
- Ingest when a certain amount of new data is recorded

Ingestion Tool

AWS Glue ETL

Ingest data on a regular basis

Ways to Ingest Data from APIs



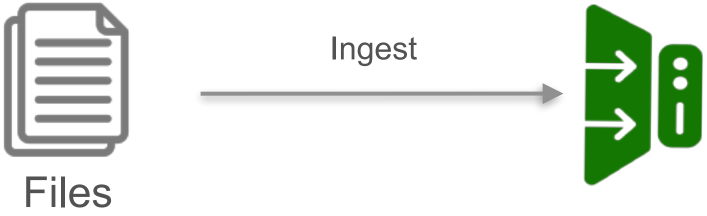
- How much can you ingest in one go?
- How frequently can you call the API?

Reading API documentation

Communicating with data owners

Writing custom API connection code

Ways to Ingest Data from Files



Manual File Download

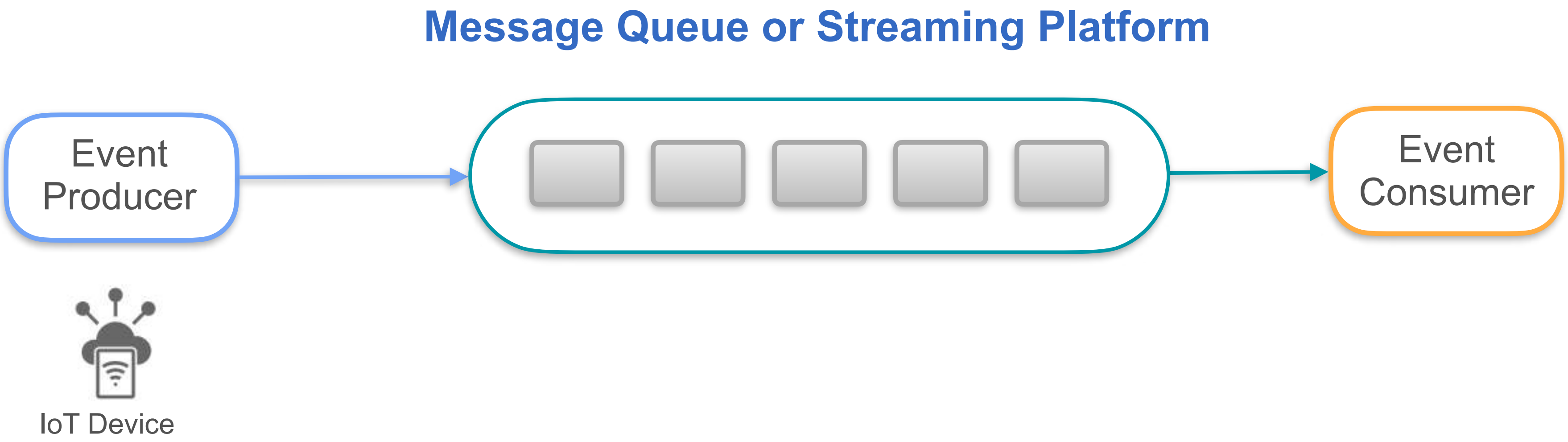
Secure File Transfer

File Transfer Protocols

SFTP: Secure File Transfer Protocol

SCP: Secure Copy Protocol

Ways to Ingest Data from Streaming Systems



DeepLearning.AI

Batch Ingestion

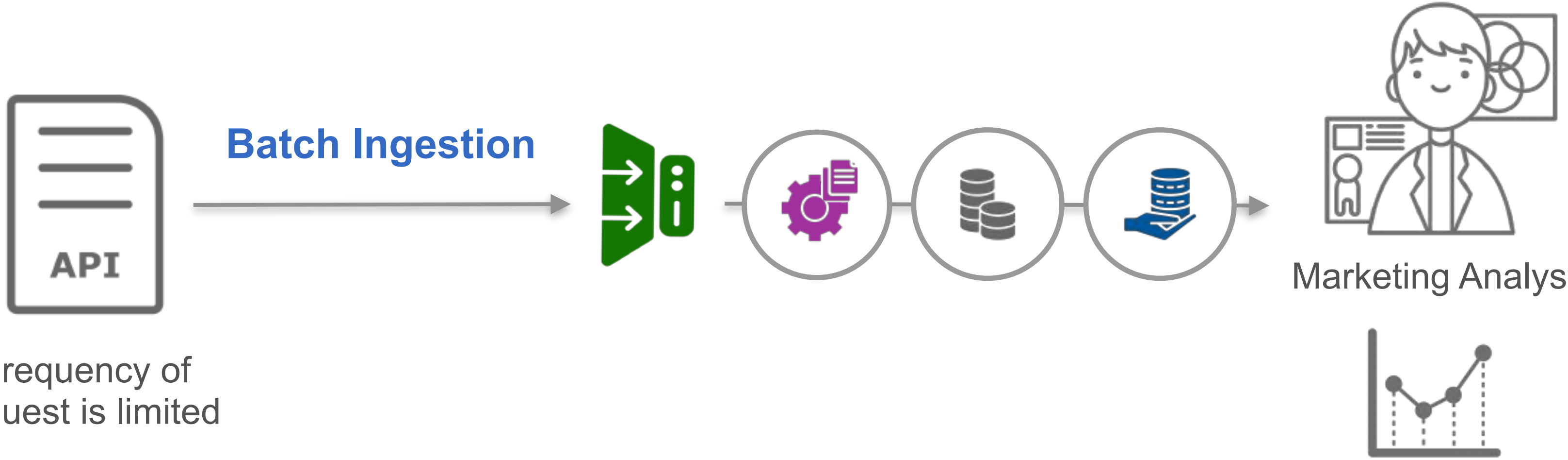
Conversation with a Marketing Analyst

DeepLearning.AI

Batch Ingestion

ETL vs. ELT

Goals of the Marketing Analyst



Batch Ingestion

Frequency of request is limited

Marketing Analyst

Analysis of Historical Trends

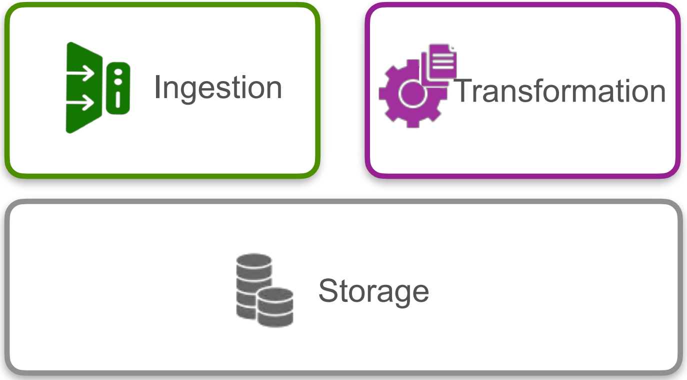
No need for real-time analysis

Batch Ingestion Patterns

ETL **Extract - Transform - Load**

Tradeoffs

ELT **Extract - Load - Transform**



Ingestion

Transformation

Storage

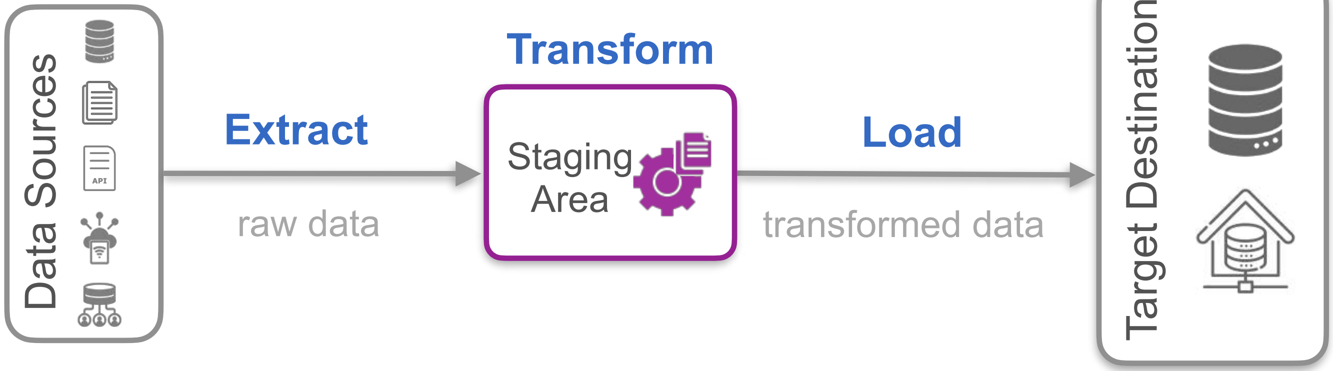
Batch Ingestion Patterns

ETL Extract - Transform - Load

ELT Extract - Load - Transform

Batch Ingestion Patterns

ETL

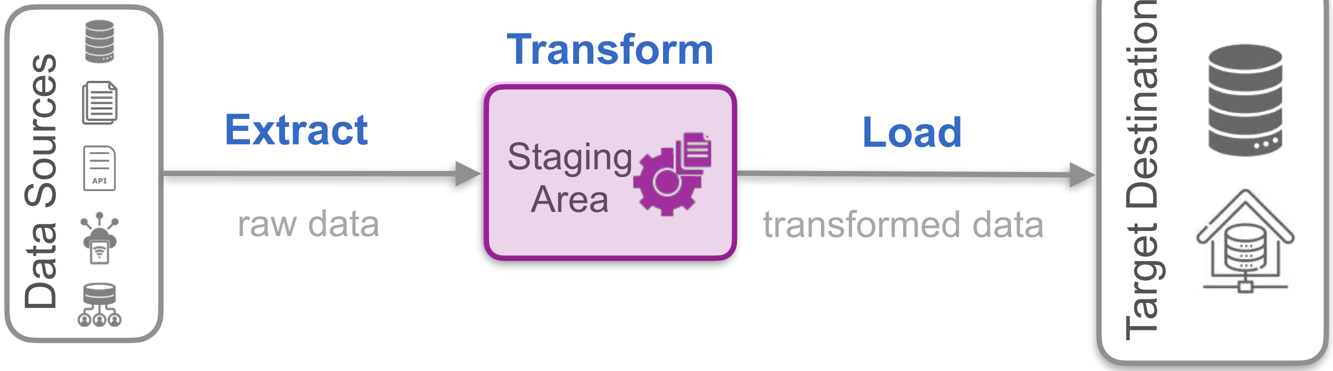


ELT

Extract - Load - Transform

Batch Ingestion Patterns

ETL



ELT

Extract - Load - Transform

Emergence of Cloud Storage Systems

Early 2010s: Highly scalable cloud storage

Data Lake

built on top of object storage

Cloud Data Warehouse

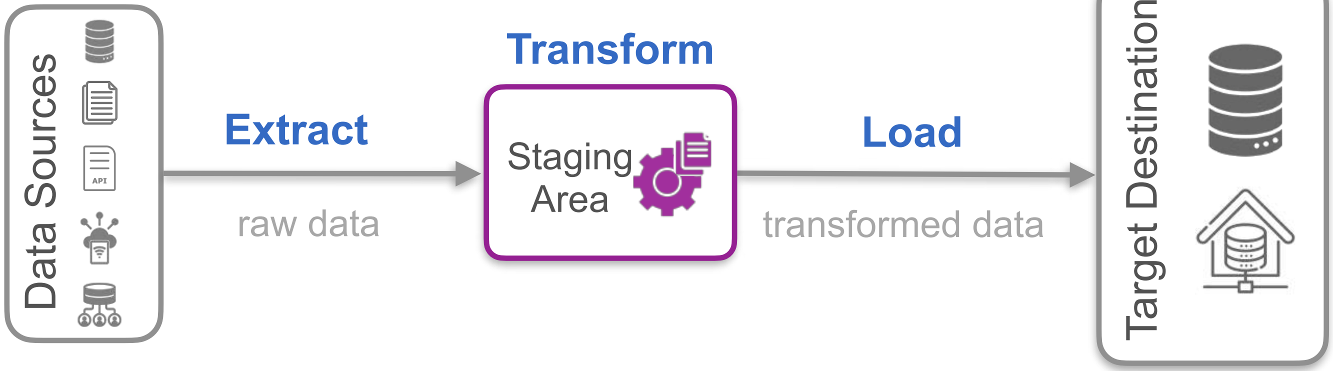
Amazon Redshift

snowflake

- Store enormous amounts of data for relatively cheap
- Perform data transformations directly in the data warehouse

Batch Ingestion Patterns

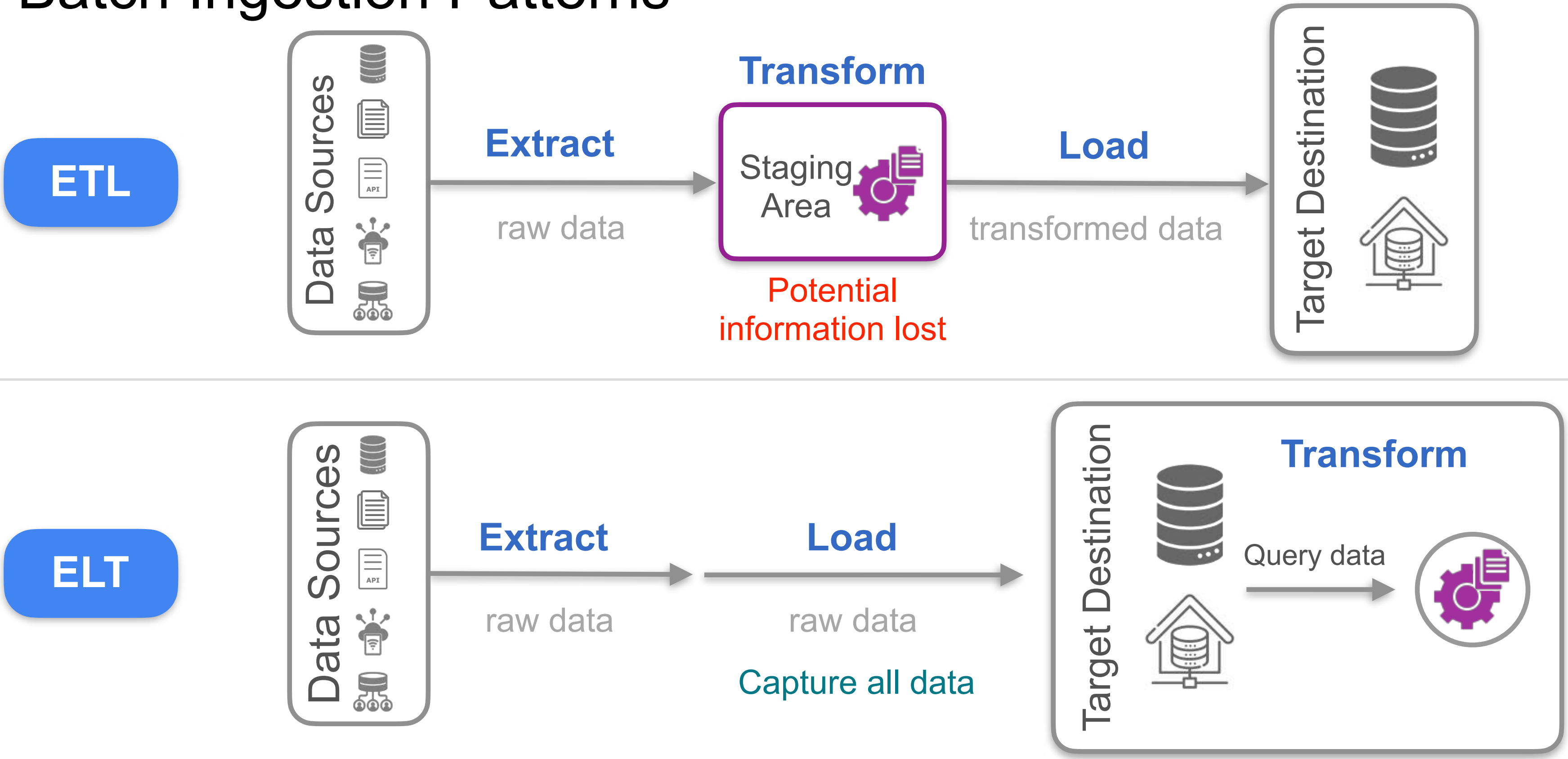
ETL



ELT

Extract - Load - Transform

Batch Ingestion Patterns



ETL

Data Sources

Extract

raw data

Transform

Staging Area

Potential information lost

Load

transformed data

Target Destination

ELT

Data Sources

Extract

raw data

Load

raw data

Capture all data

Target Destination

Transform

Query data

Advantages of Extract-Load-Transform

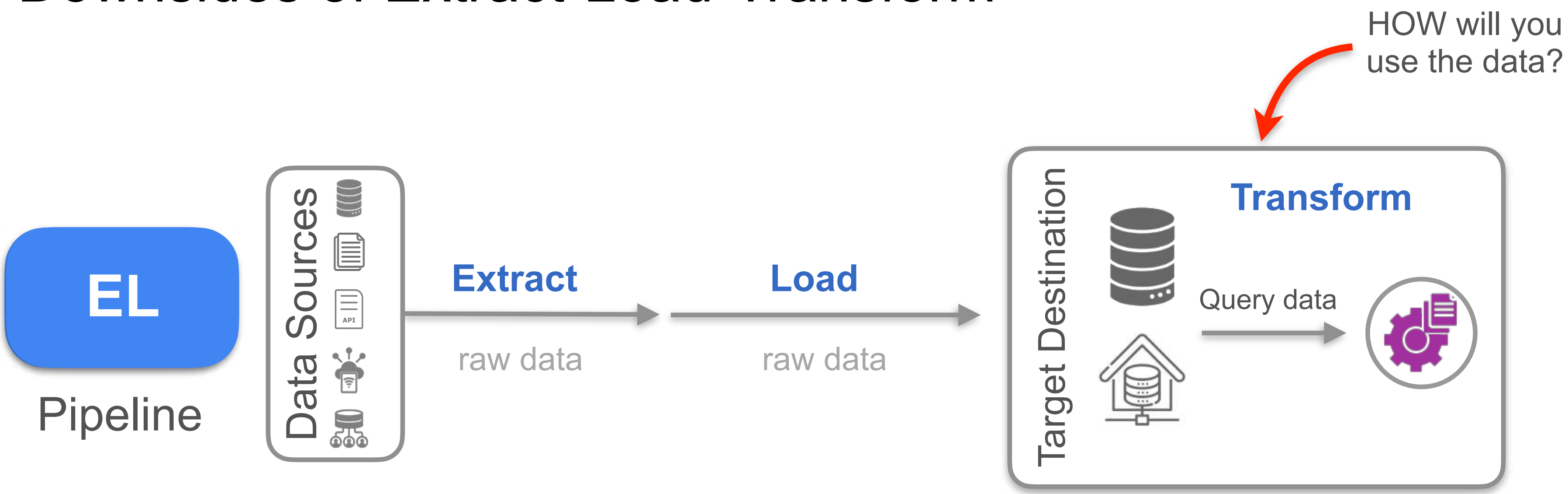
It is faster to implement.

It makes data available more quickly to end users.

Transformations can still be done efficiently.

You can decide later to adopt different transformations.

Downsides of Extract-Load-Transform



Data Swamp

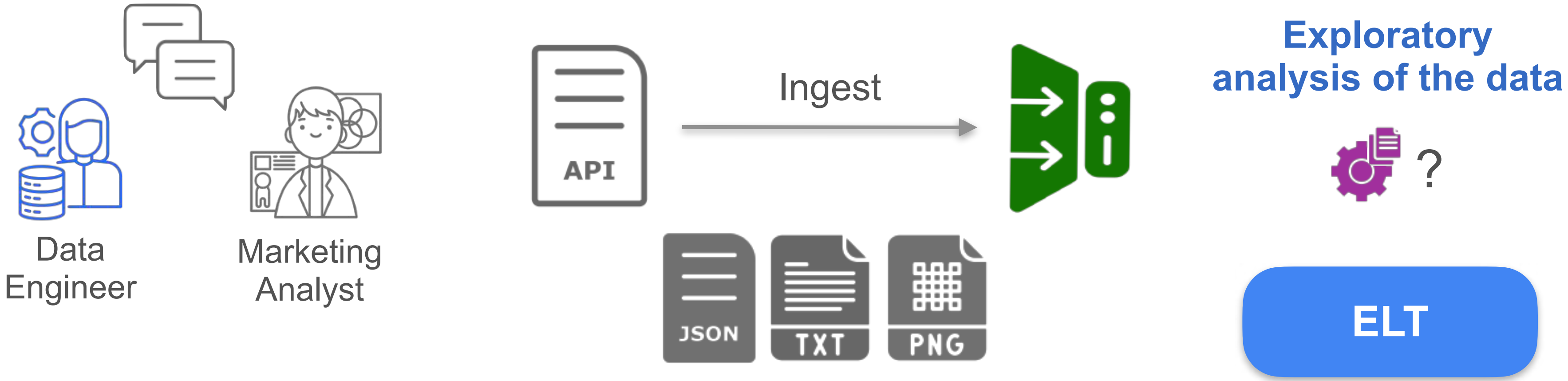
Data has become unorganized, unmanageable, and essentially useless.

Downsides of Extract-Load-Transform

Data Swamp

Video by Adobe Stock (paid license)

Conversation with the Marketing Analyst

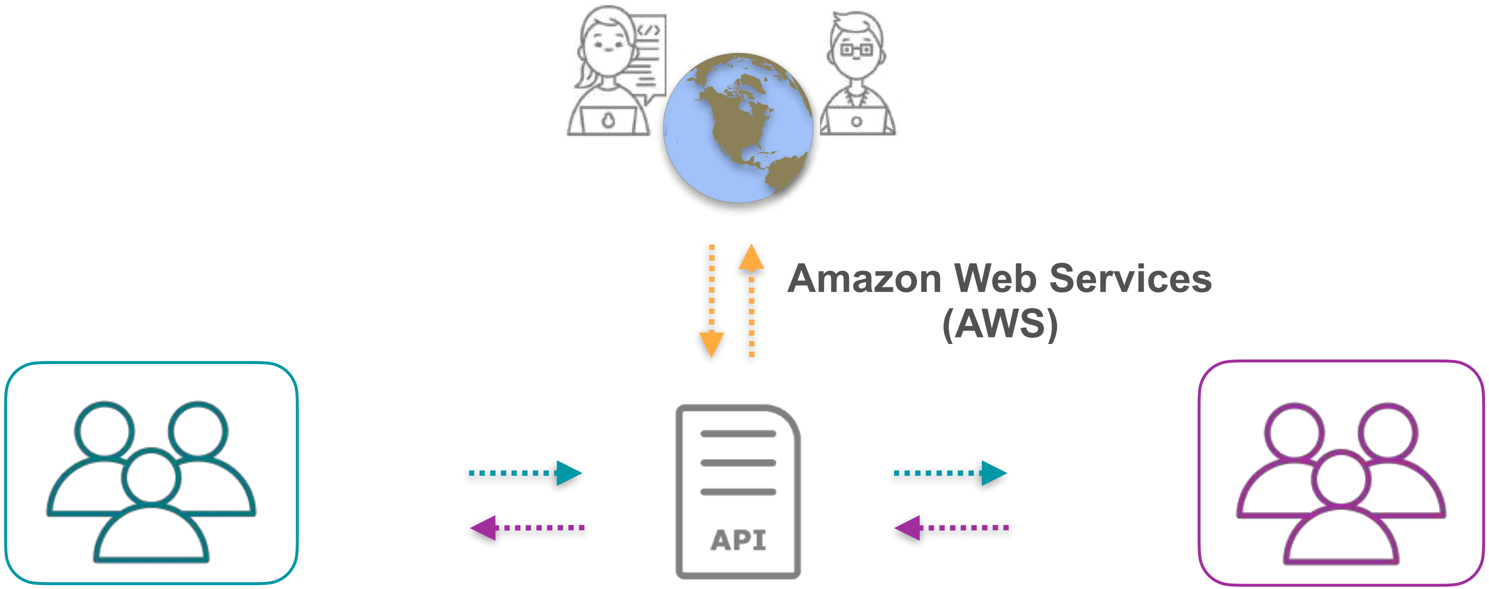


DeepLearning.AI

Batch Ingestion

REST API

API Mandate



Amazon Web Services
(AWS)

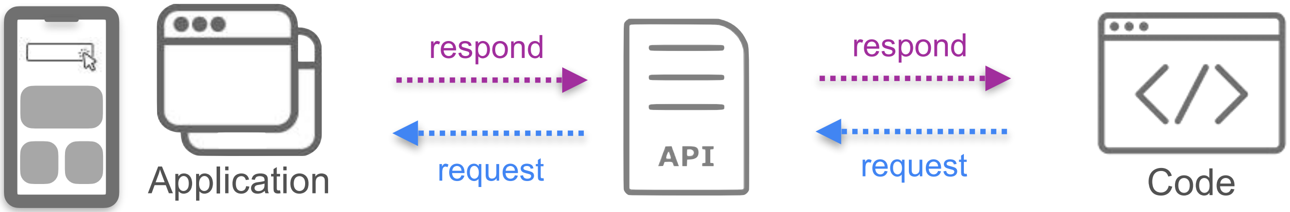
Application
Programming Interface

Stable & Predictable

What is an API?

API

A set of rules and specifications that allows you to programmatically communicate and exchange data with an application.

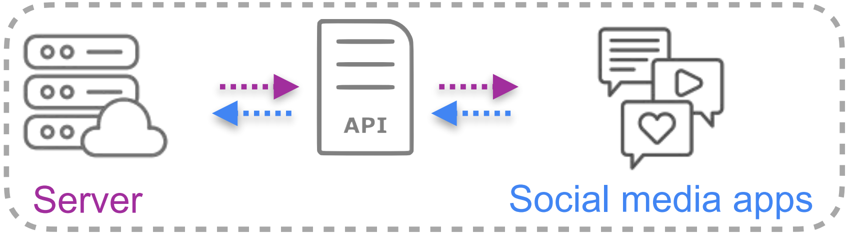


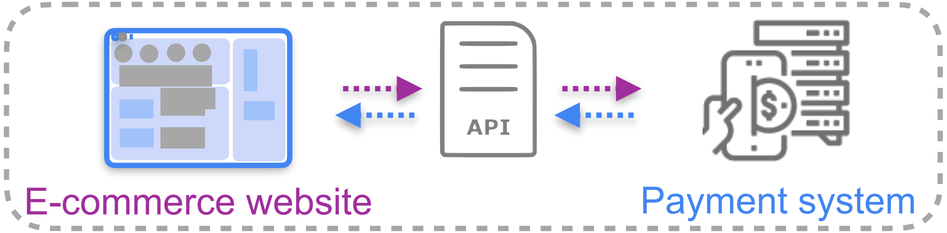
Built into a wide range of software applications

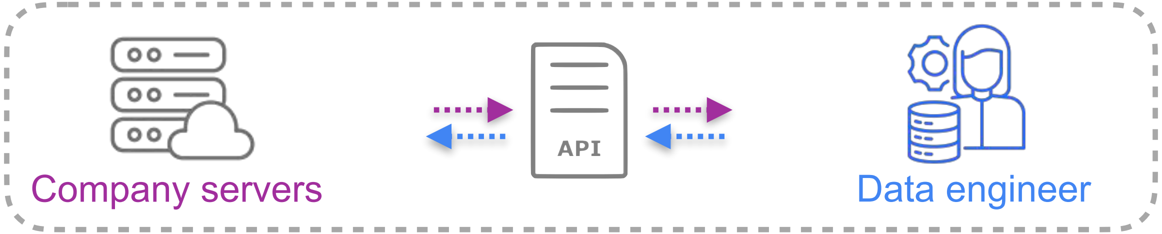
What is an API?

API

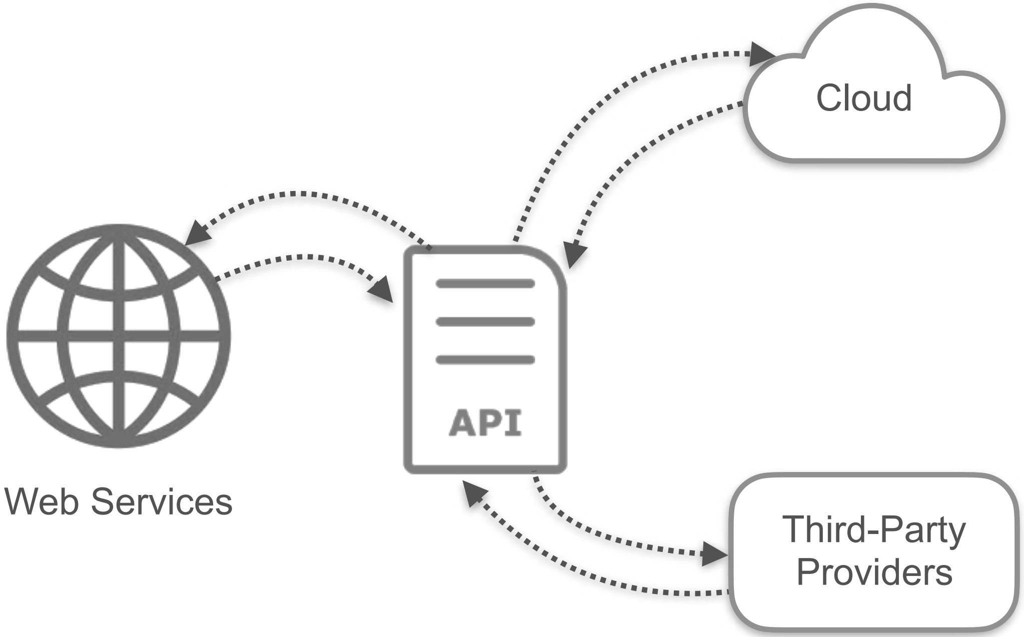
A set of rules and specifications that allows you to programmatically communicate and exchange data with an application.







What is an API?



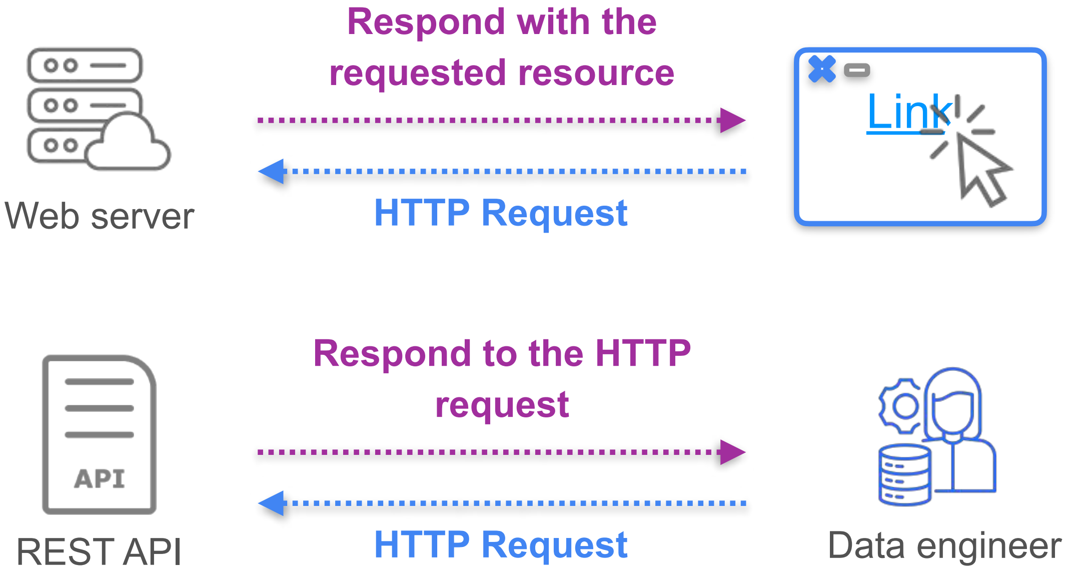
API Features

- Metadata
- Documentation
- Authentication
- Error handling

REST API

REST API

Representational State Transfer API
Use Hypertext Transfer Protocol (HTTP) as the basis for communication


Respond with the requested resource
Link
Web server
HTTP Request
Respond to the HTTP request
API
REST API
HTTP Request
Data engineer

DeepLearning.AI

Lab Walkthrough

Batch Processing to Get Data From an API

Batch Processing from an API

Upcoming Lab

Spotify

- Extract data from the Spotify API
- Explore what pagination means
- Send an API request that requires authorization

In this video,

- Go through some API concepts
- Give you an overview of the lab tasks

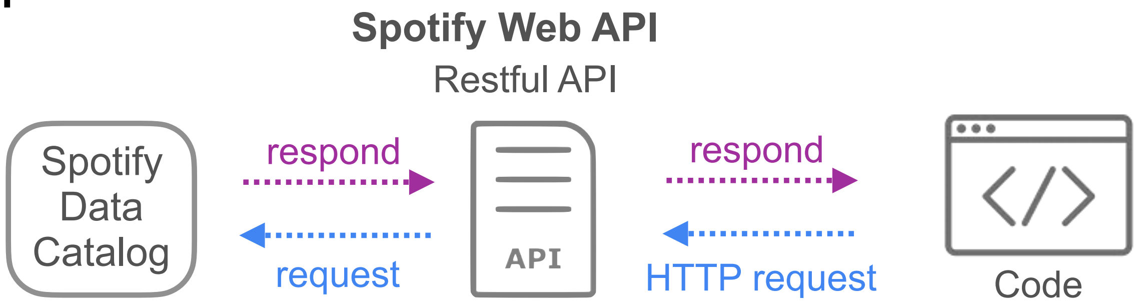
What you need

- Spotify account https://developer.spotify.com/

  When working with an API, it’s very common that you’ll have to sign up for an account

- Spotify Documentation https://developer.spotify.com/documentation/web-api

API Concepts



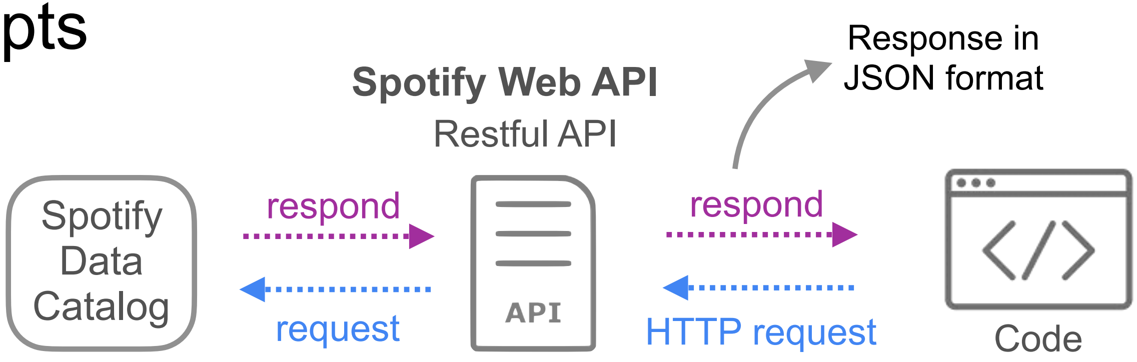
Each resource is represented by an **endpoint**.

Resource

- Music
- Artists
- Albums
- Tracks
- Playlist

HTTP Request	Action
GET	Retrieve a resource
POST	Create a resource
PUT	Change/Replace a resource
Delete	Delete a resource

API Concepts



Resource

- Music
- Artists
- Albums
- Tracks
- Playlist

Each resource is represented by an **endpoint**.

HTTP Request	Action
GET	Retrieve a resource
POST	Create a resource
PUT	Change/Replace a resource
Delete	Delete a resource

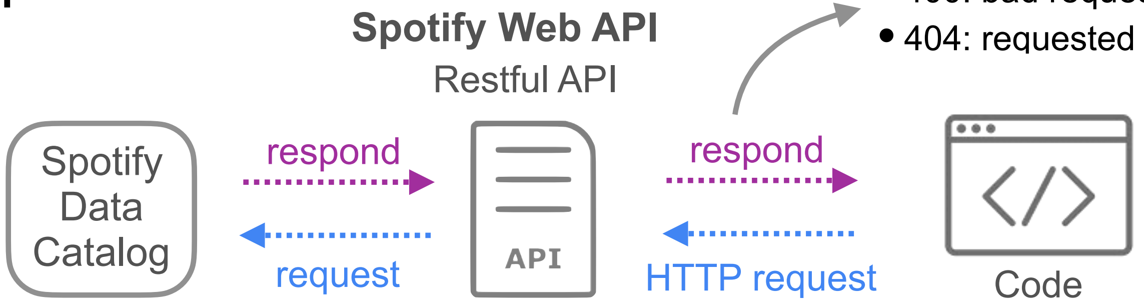
API Concepts

Error object containing a status code:

- 400: bad request
- 404: requested resource not found

Spotify Web API

Restful API



Each resource is represented by an **endpoint**.

Resource

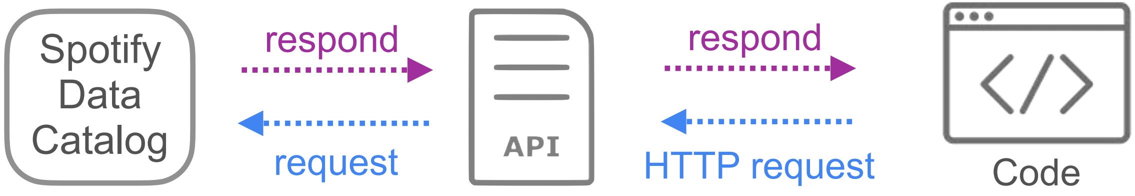
- Music
- Artists
- Albums
- Tracks
- Playlist

HTTP Request	Action
GET	Retrieve a resource
POST	Create a resource
PUT	Change/Replace a resource
Delete	Delete a resource

API Concepts

Spotify Web API

Restful API



: Endpoint + Access token

Each resource is represented by an **endpoint**.

Resource

- Music
- Artists
- Albums
- Tracks
- Playlist

Access token: string that contains the permissions to access a given resource. *(valid for 1 hour)*

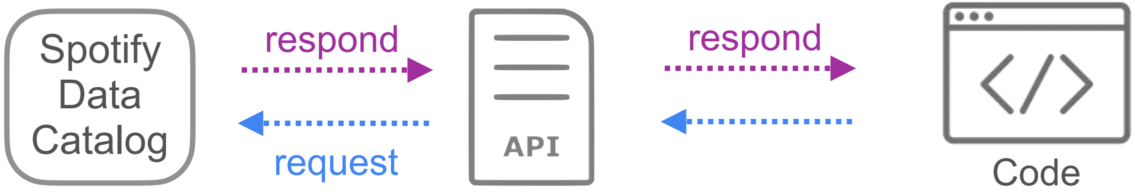
- create a Spotify account
- get a client ID and a client secret and use them to generate the access token *(provided with the code)*

https://developer.spotify.com/documentation/web-api/concepts/authorization

API Concepts

Spotify Web API

Restful API

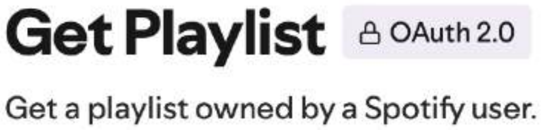


https://developer.spotify.com/documentation/web-api s token

Resource

- Music
- Artists
- Albums
- -
- F

Each resource is represented by an **endpoint**.



Get Playlist

OAuth 2.0

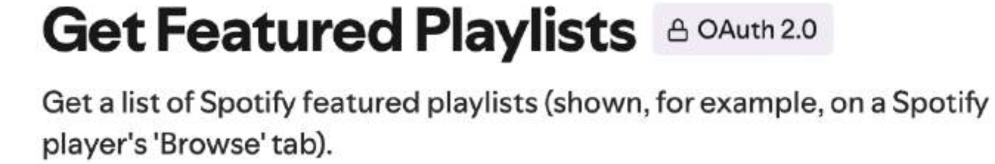
Get a playlist owned by a Spotify user.

ontains the permissions to access

hour)

: secret and use them to generate

de)



Get Featured Playlists

OAuth 2.0

Get a list of Spotify featured playlists (shown, for example, on a Spotify player's 'Browse' tab).

https authorization

Pagination

Extract the items chunk by chunk.

- Using offset and limit

```
https://api.spotify.com/v1/me/shows?offset=0&limit=20
```

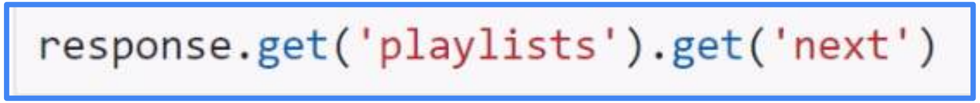
```
https://api.spotify.com/v1/me/shows?offset=20&limit=20
```

```
https://api.spotify.com/v1/me/shows?offset=40&limit=20
```

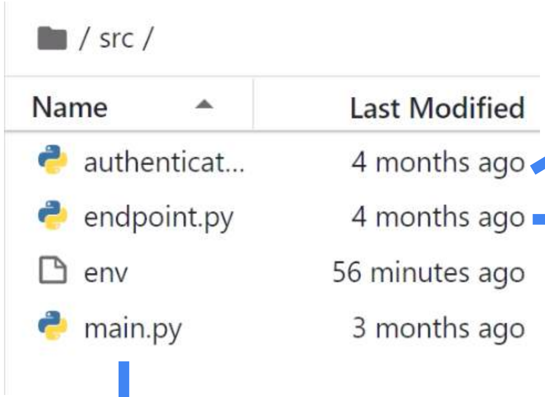
```
https://api.spotify.com/v1/me/shows?offset=60&limit=20
```

```
https://api.spotify.com/v1/me/shows?offset=80&limit=20
```

- Using the next field



```
response.get('playlists').get('next')
```



/ src /

Name	Last Modified
authenticat...	4 months ago
endpoint.py	4 months ago
env	56 minutes ago
main.py	3 months ago

Contains the scripts of the get_token function

1. Paginated call to the endpoint “Get featured playlists”
2. Paginated call to the endpoint “Get playlist”
3. Automatically generate a new token

1. Get the ids of the featured playlists
2. Extract the track information for each playlist id

DeepLearning.AI

Streaming Ingestion

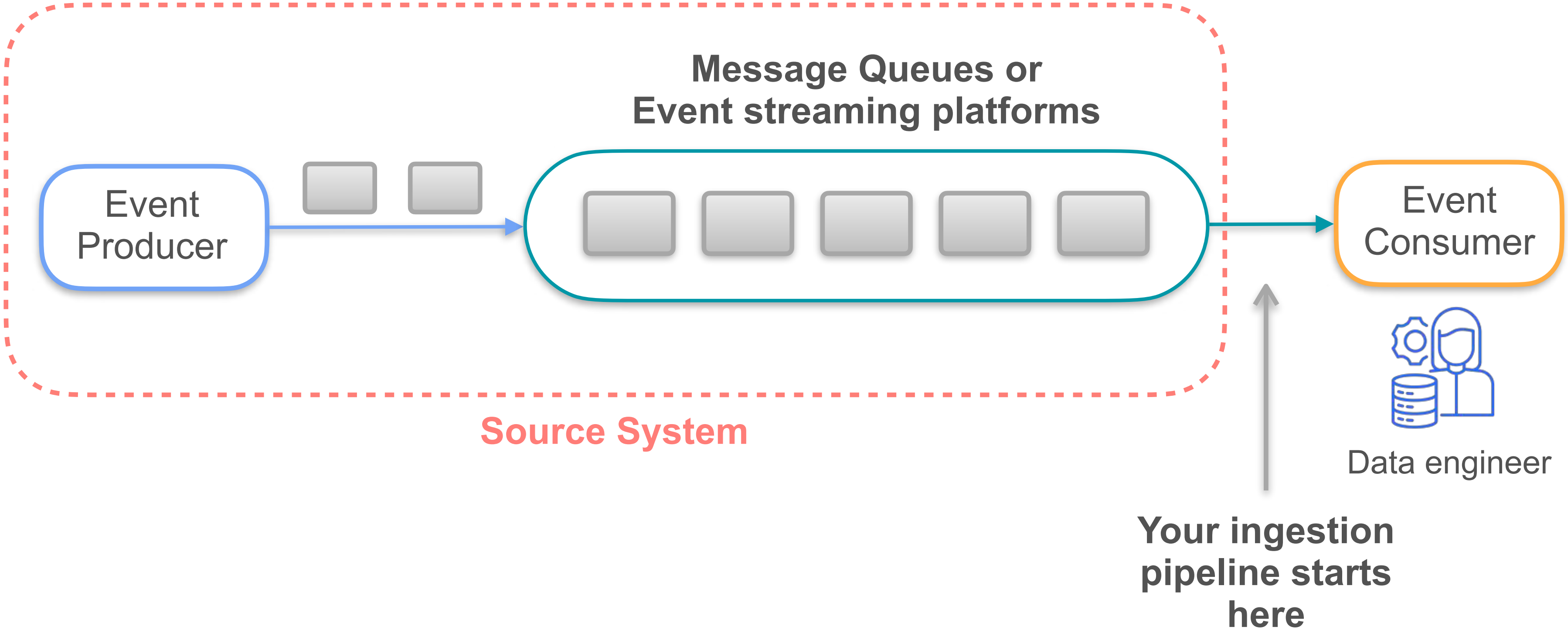
Conversation with a Software Engineer

DeepLearning.AI

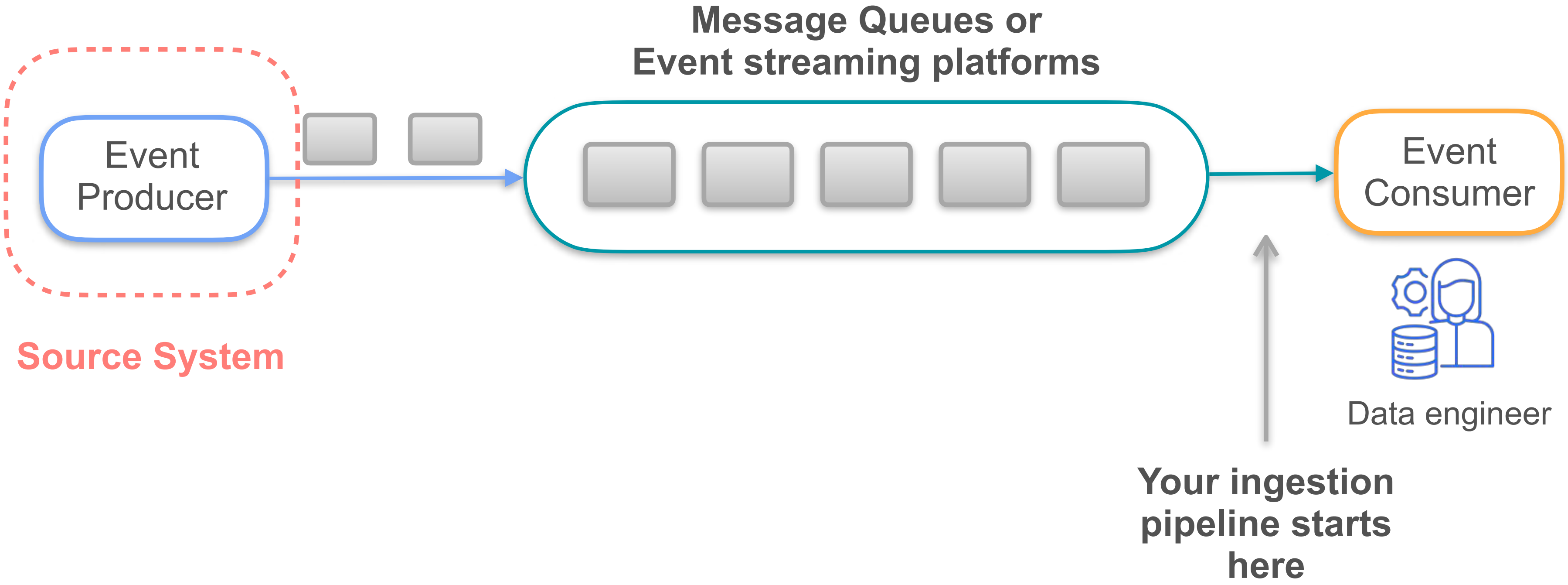
Streaming Ingestion

Streaming Ingestion Details

Streaming Systems



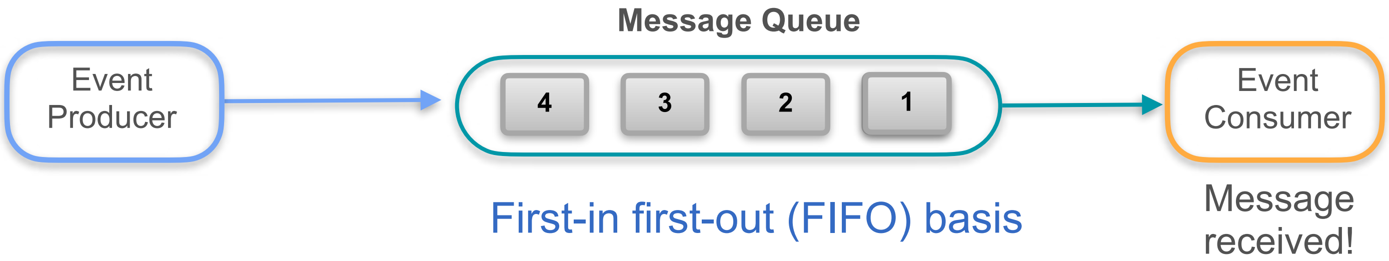
Streaming Systems



Message Queue

Event Streaming Platform

Message Queue A buffer used to deliver messages asynchronously



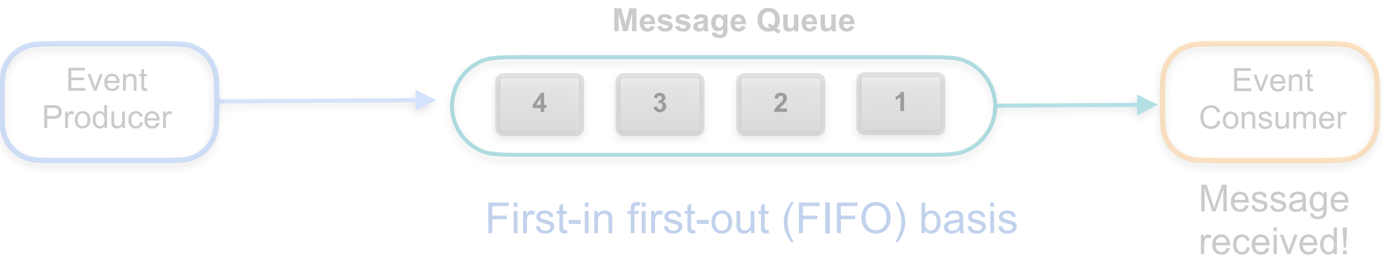
First-in first-out (FIFO) basis

Message received!

Event Streaming Platform

Message Queue

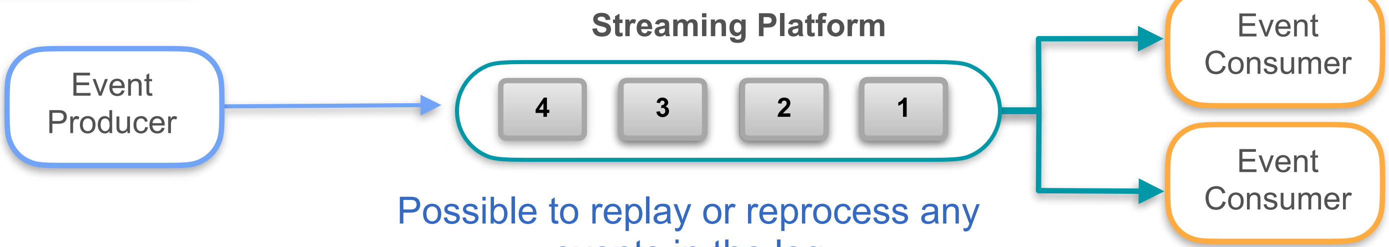
A buffer used to deliver messages asynchronously



First-in first-out (FIFO) basis

Event Streaming Platform

Append-only persistent log



Possible to replay or reprocess any events in the log

Event Streaming Platforms

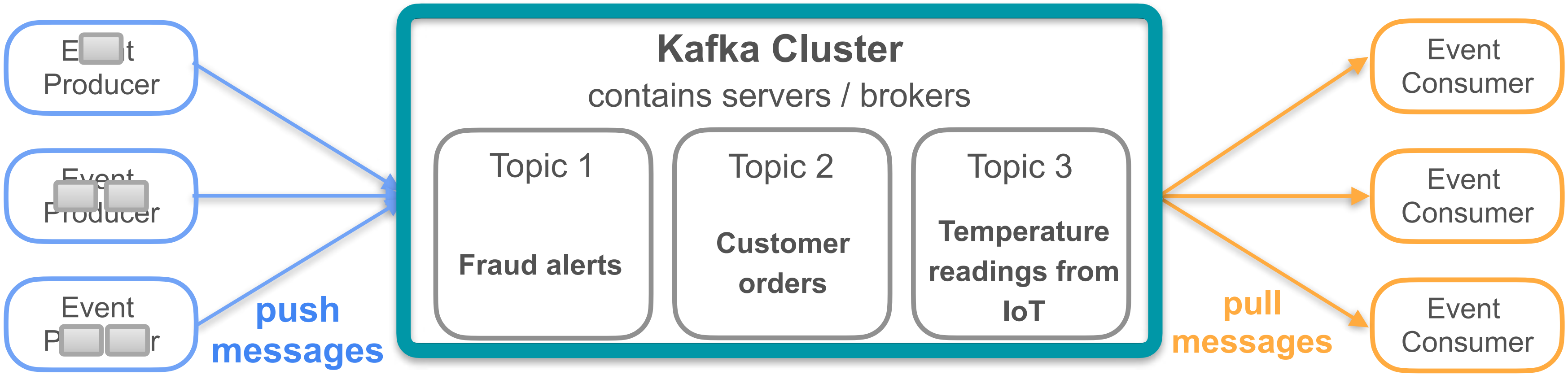
In this week's lab:

Amazon Kinesis
Data Streams

In this video:

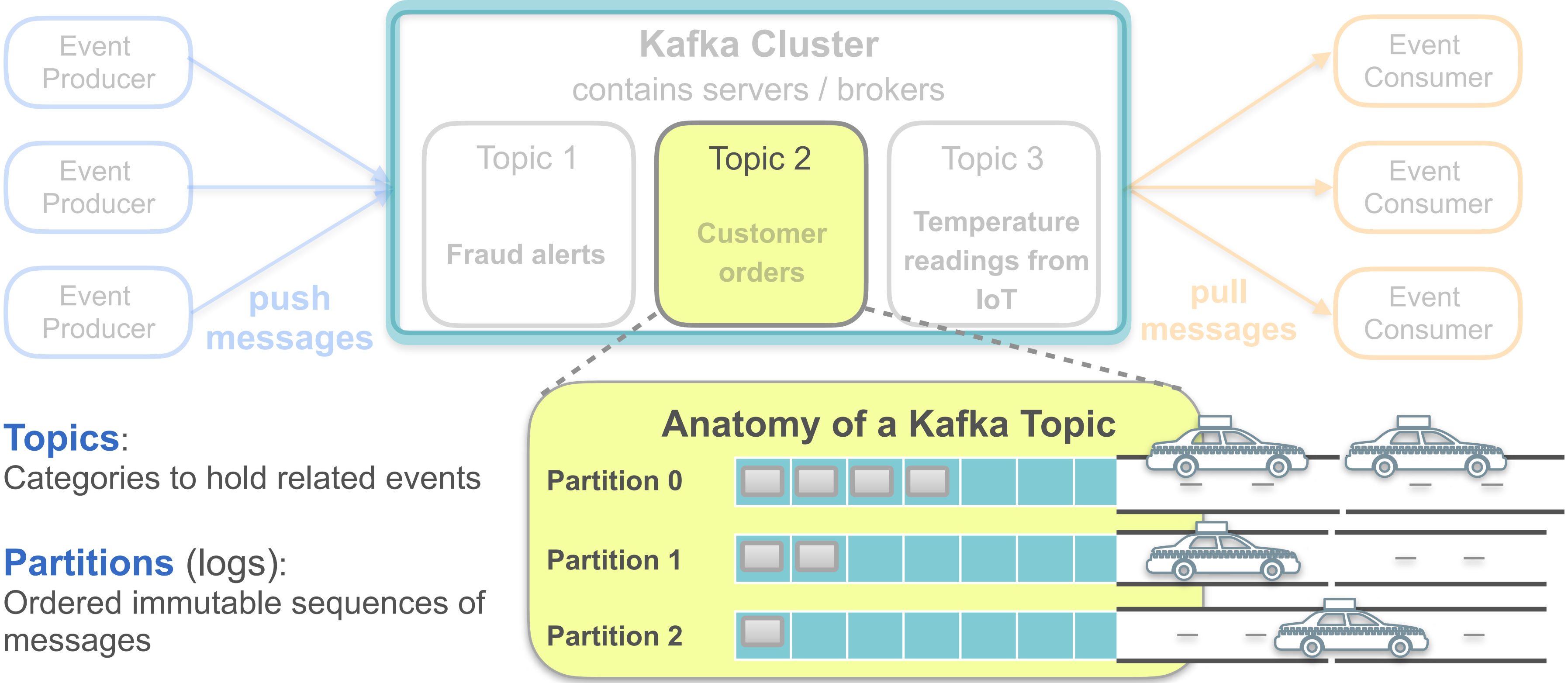
Apache Kafka

kafka Open-source event streaming platform



Topics:
Categories to hold related events

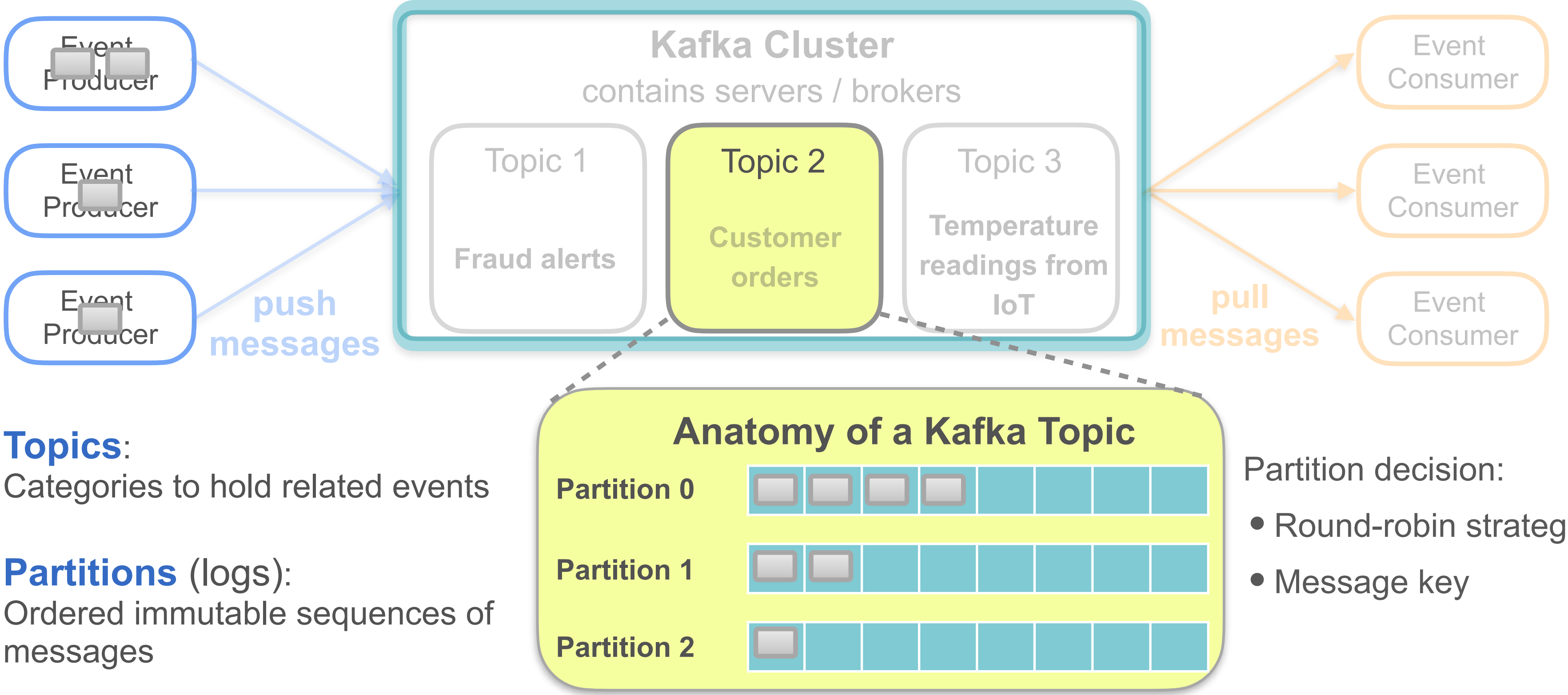
kafka Open-source event streaming platform



Topics:
Categories to hold related events

Partitions (logs):
Ordered immutable sequences of messages

kafka Open-source event streaming platform



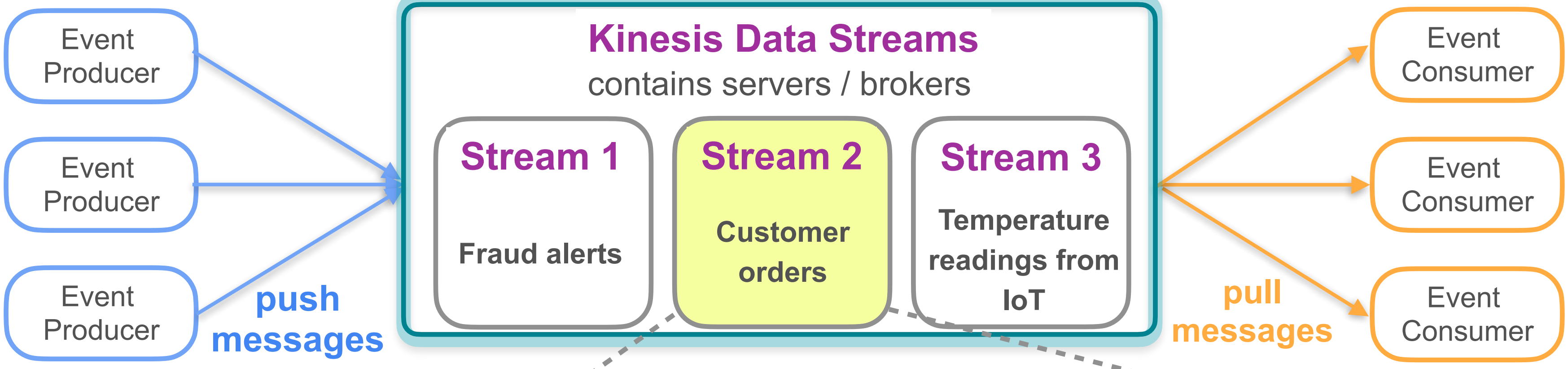
Topics:
Categories to hold related events

Partitions (logs):
Ordered immutable sequences of messages

Partition decision:

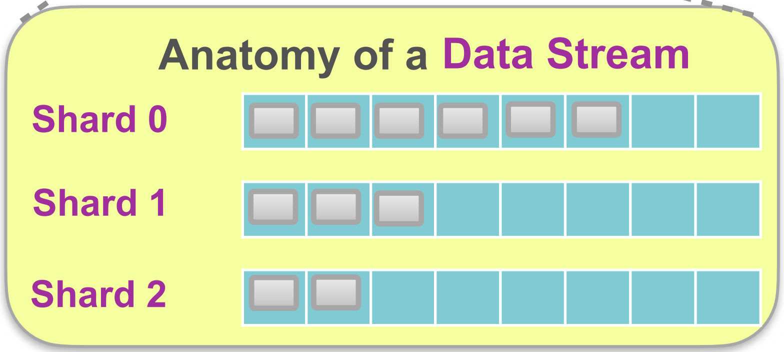
- Round-robin strategy
- Message key

Amazon Kinesis Data Stream

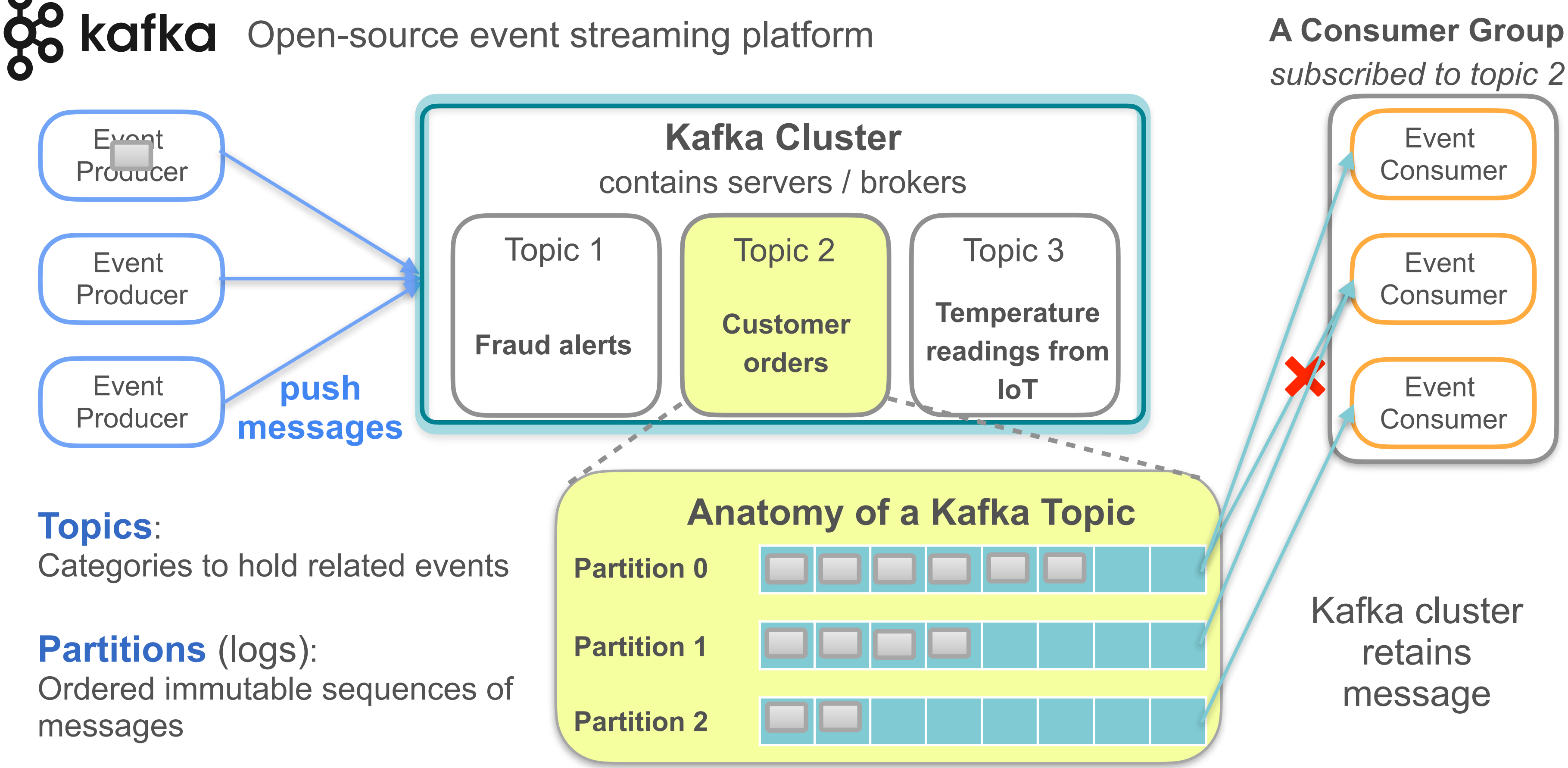


Stream:
Categories to hold related events

Shard (logs):
Ordered immutable sequences of messages



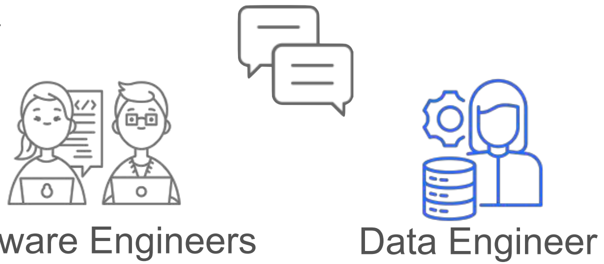
kafka Open-source event streaming platform

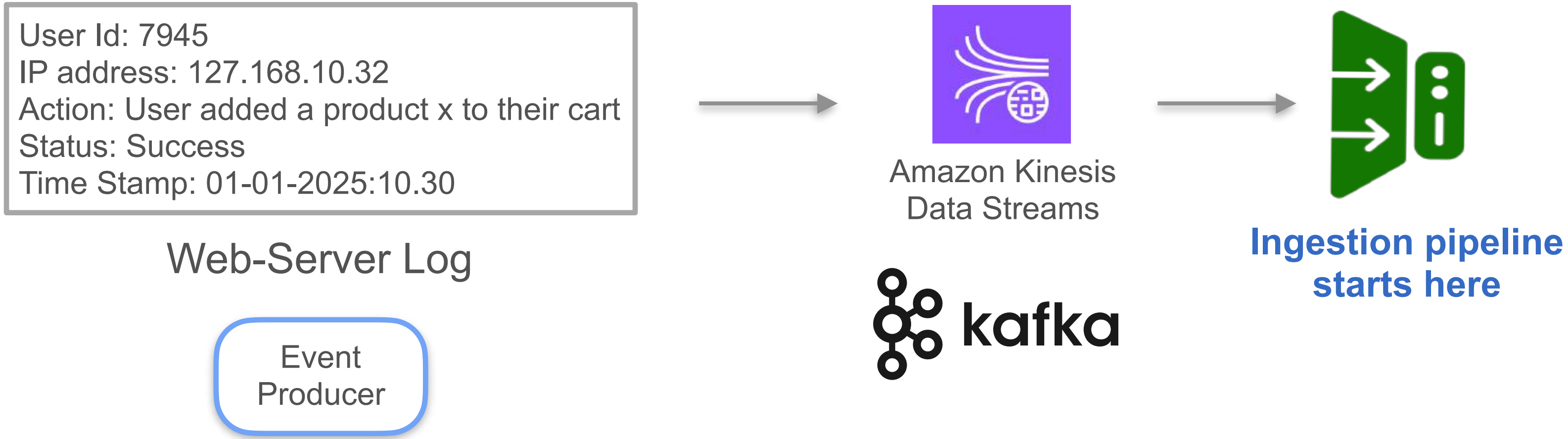


Topics:
Categories to hold related events

Partitions (logs):
Ordered immutable sequences of messages

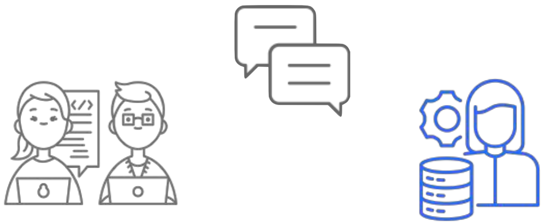
Conversation with the Software Engineer





User Id: 7945
IP address: 127.168.10.32
Action: User added a product x to their cart
Status: Success
Time Stamp: 01-01-2025:10.30

Conversation with the Software Engineer



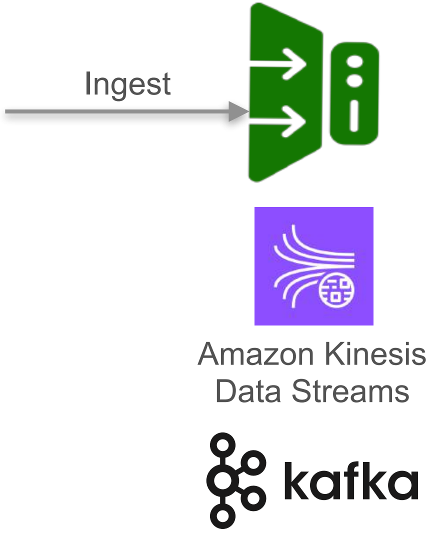
Software Engineers

Data Engineer

User Id: 7945
IP address: 127.168.10.32
Action: User added a product x to their cart
Status: Success
Time Stamp: 01-01-2025:10.30

Web-Server Log

Event Producer



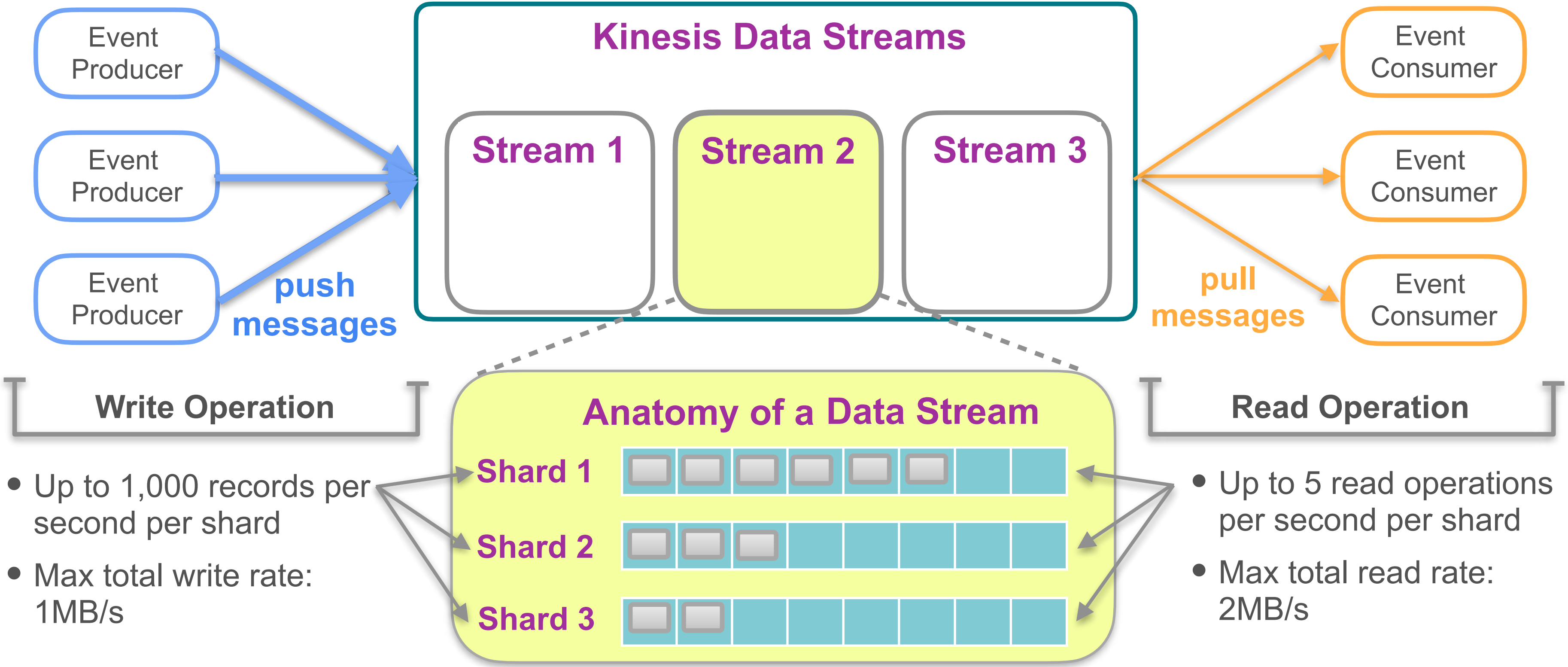
Ingestion pipeline starts here

Amazon Kinesis Data Streams

DeepLearning.AI

Streaming Ingestion

Amazon Kinesis Data Streams Details



Write Operation

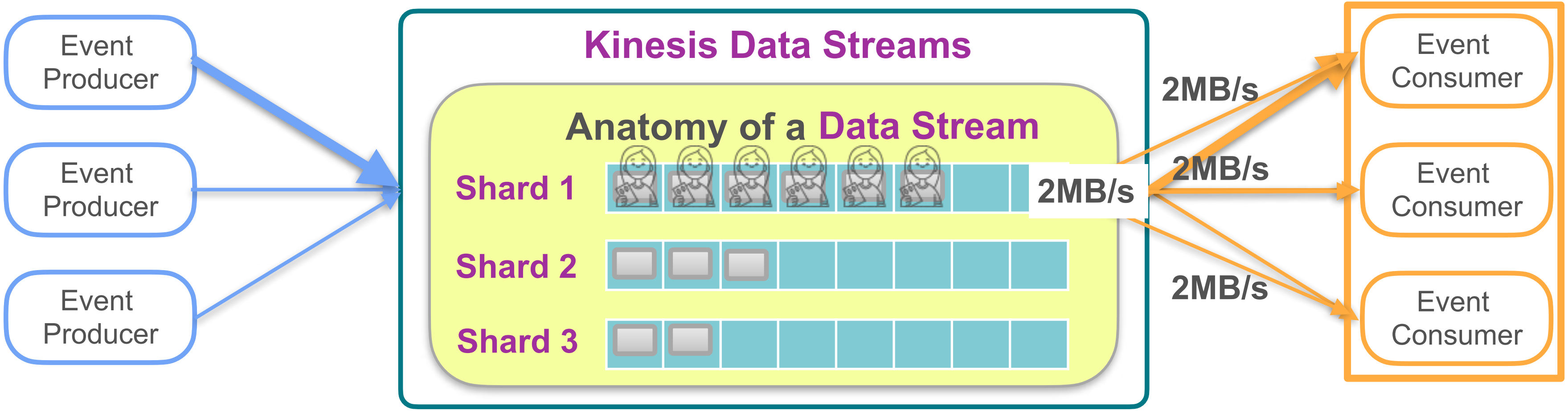
- Up to 1,000 records per second per shard
- Max total write rate: 1MB/s

Read Operation

- Up to 5 read operations per second per shard
- Max total read rate: 2MB/s

Size and rate of write and read operations

Kinesis in “on-demand” Mode	Kinesis in “Provisioned” Mode
• Automatically manage the scaling of the shards up or down as needed • Only charged for what you use • More convenient from an operational perspective	• Specify the number of shards necessary for your application based on the expected write and read request rate • Manually add more shards or re-shard when needed • A good fit if… ◦ you have predictable application traffic ◦ you are able to control your costs more carefully



Event Producer

Event Producer

Event Producer

Kinesis Data Streams

Anatomy of a Data Stream

Shard 1

Shard 2

Shard 3

2MB/s

2MB/s

2MB/s

2MB/s

Event Consumer

Event Consumer

Event Consumer

Data Record

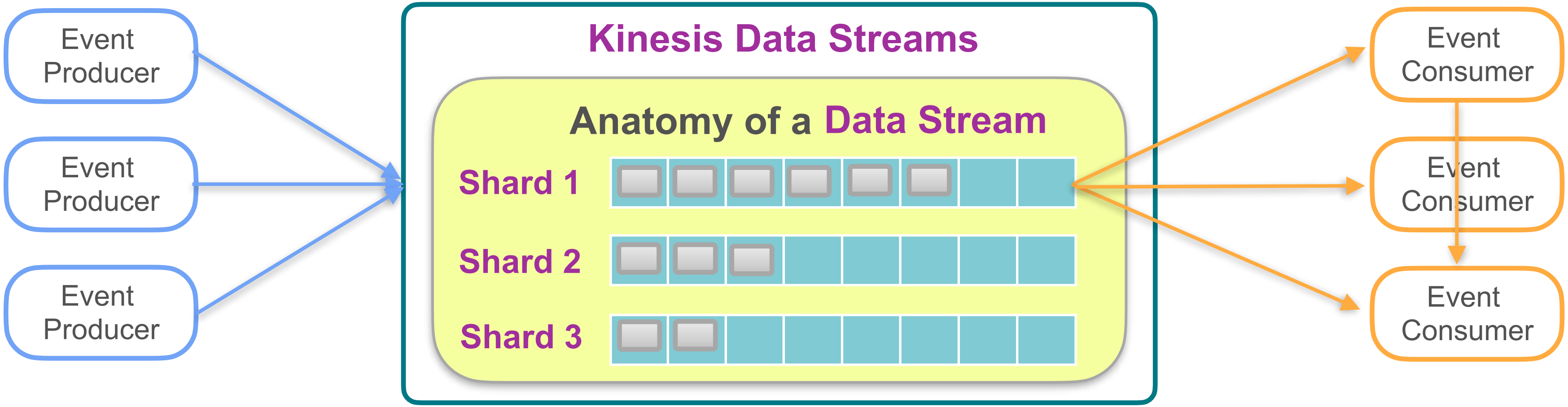
Partition Key customerID *12567910* → Used to determine which shard the data record is placed into

Sequence Number

Binary Large Object (BLOB)

Shared Fan-Out When consumers share a shard's read capacity

Enhanced Fan-Out When consumers are able to read at the full read capacity of the shard



AWS Lambda

Amazon Managed Service for Apache Flink

AWS Glue

Amazon Data Firehose

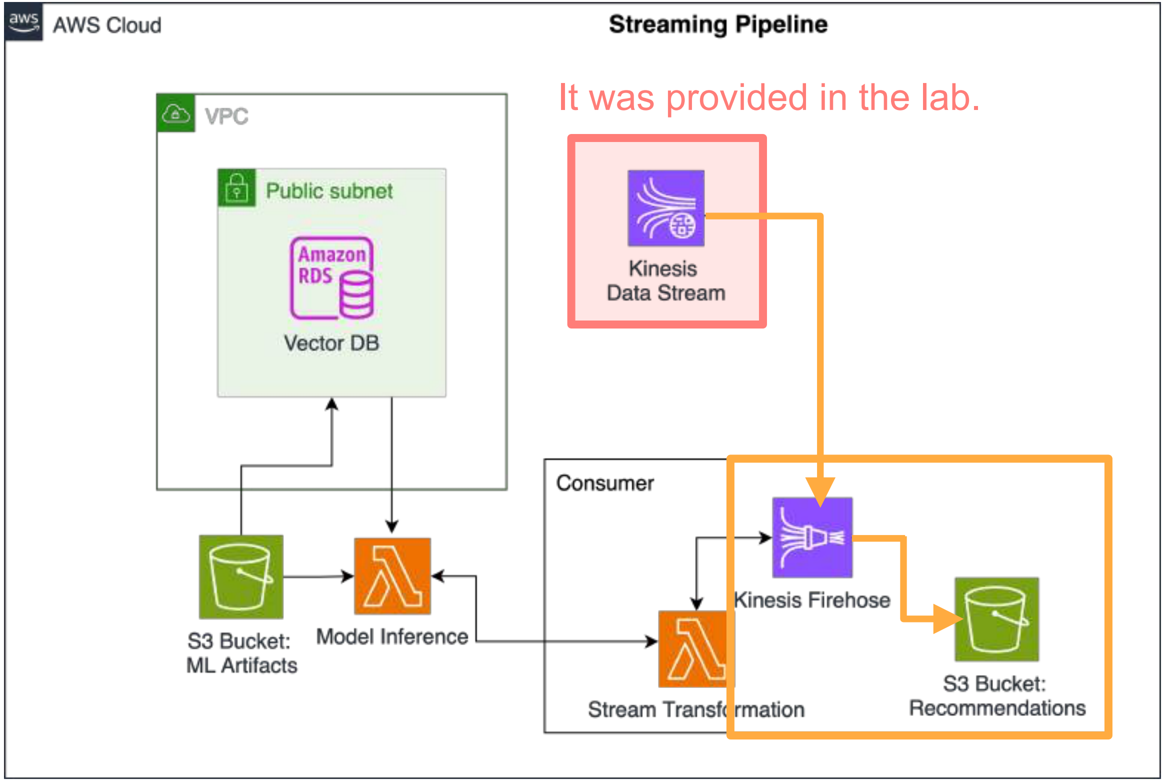
Amazon Kinesis Client Library (KCL)

DeepLearning.AI

Lab Walkthrough

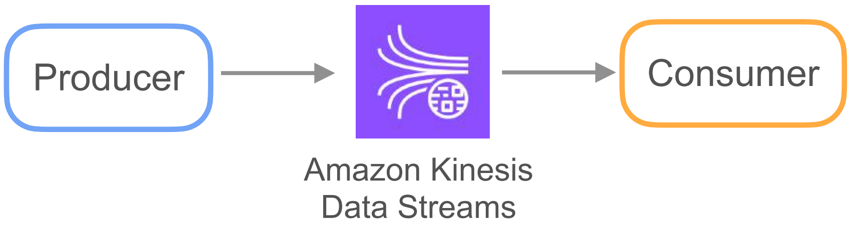
Streaming Ingestion

Course 1 Lab

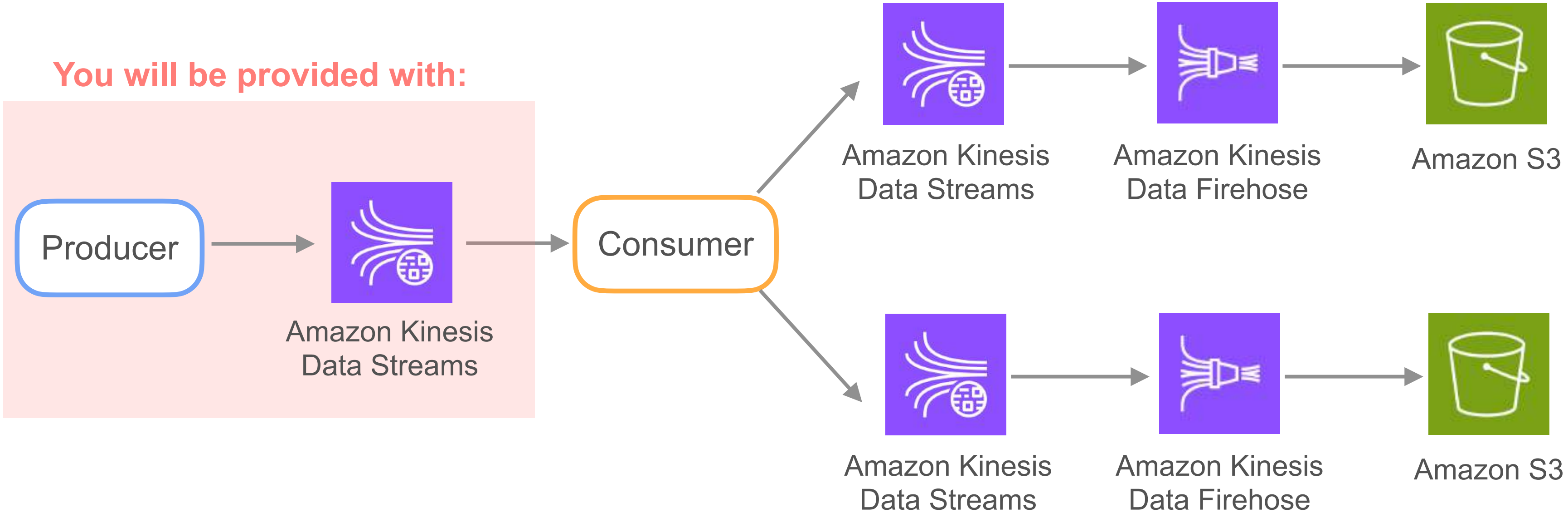


It was provided in the lab.

Part 1

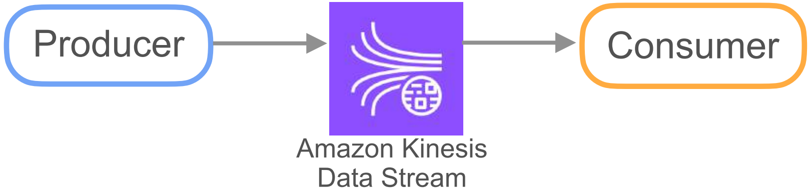


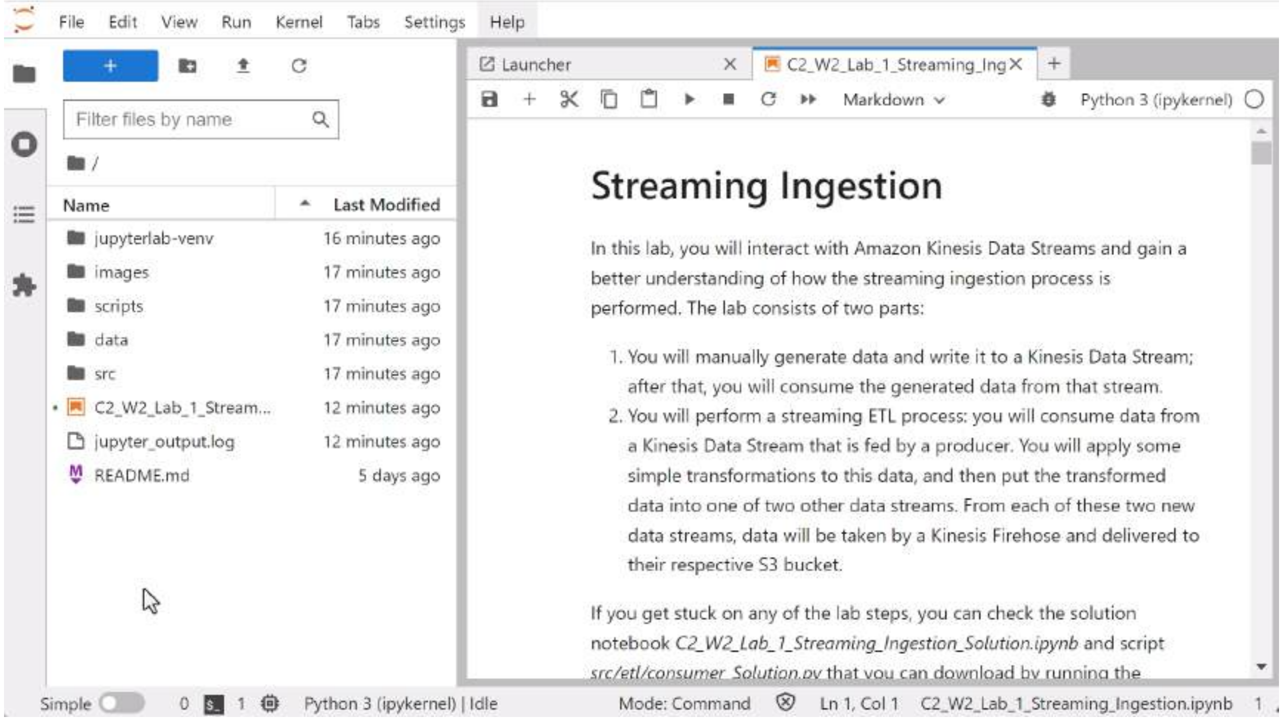
Part 2


You will be provided with:

Producer
Amazon Kinesis Data Streams
Consumer
Amazon Kinesis Data Streams
Amazon Kinesis Data Firehose
Amazon S3
Amazon Kinesis Data Streams
Amazon Kinesis Data Firehose
Amazon S3

Part 1





Name	Last Modified
jupyterlab-venv	16 minutes ago
images	17 minutes ago
scripts	17 minutes ago
data	17 minutes ago
src	17 minutes ago
C2_W2_Lab_1_Stream...	12 minutes ago
jupyter_output.log	12 minutes ago
README.md	5 days ago

Streaming Ingestion

In this lab, you will interact with Amazon Kinesis Data Streams and gain a better understanding of how the streaming ingestion process is performed. The lab consists of two parts:

1. You will manually generate data and write it to a Kinesis Data Stream; after that, you will consume the generated data from that stream.
2. You will perform a streaming ETL process: you will consume data from a Kinesis Data Stream that is fed by a producer. You will apply some simple transformations to this data, and then put the transformed data into one of two other data streams. From each of these two new data streams, data will be taken by a Kinesis Firehose and delivered to their respective S3 bucket.

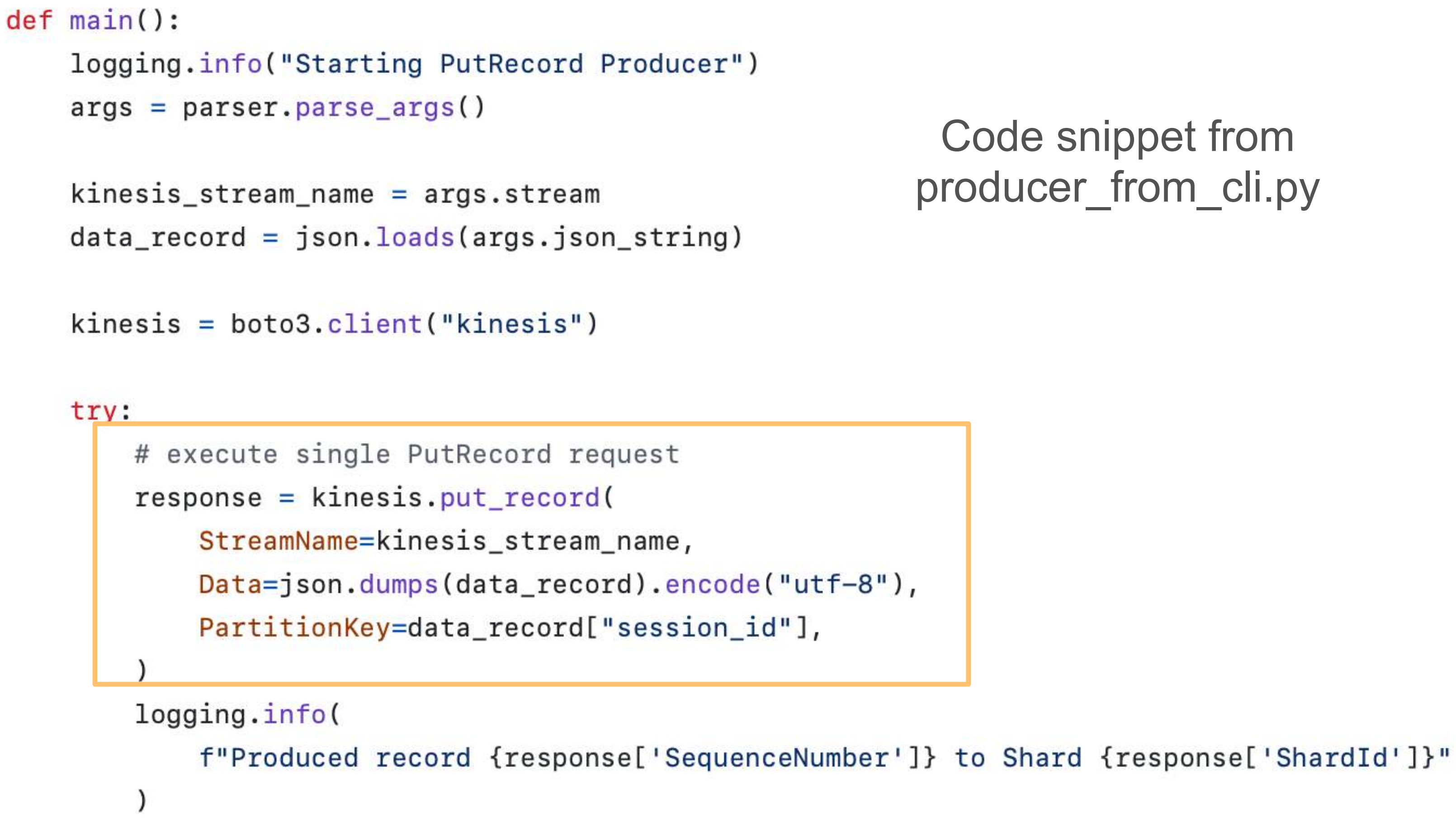
If you get stuck on any of the lab steps, you can check the solution notebook *C2_W2_Lab_1_Streaming_Ingestion_Solution.ipynb* and script *src/etl/consumer_Solution.py* that you can download by running the

Part 1

produ

- write
- uses
- can

```
pytho
```



```
def main():
    logging.info("Starting PutRecord Producer")
    args = parser.parse_args()

    kinesis_stream_name = args.stream
    data_record = json.loads(args.json_string)

    kinesis = boto3.client("kinesis")

    try:
        # execute single PutRecord request
        response = kinesis.put_record(
            StreamName=kinesis_stream_name,
            Data=json.dumps(data_record).encode("utf-8"),
            PartitionKey=data_record["session_id"],
        )
        logging.info(
            f"Produced record {response['SequenceNumber']} to Shard {response['ShardId']}"
        )
```

Code snippet from producer_from_cli.py

Part 1

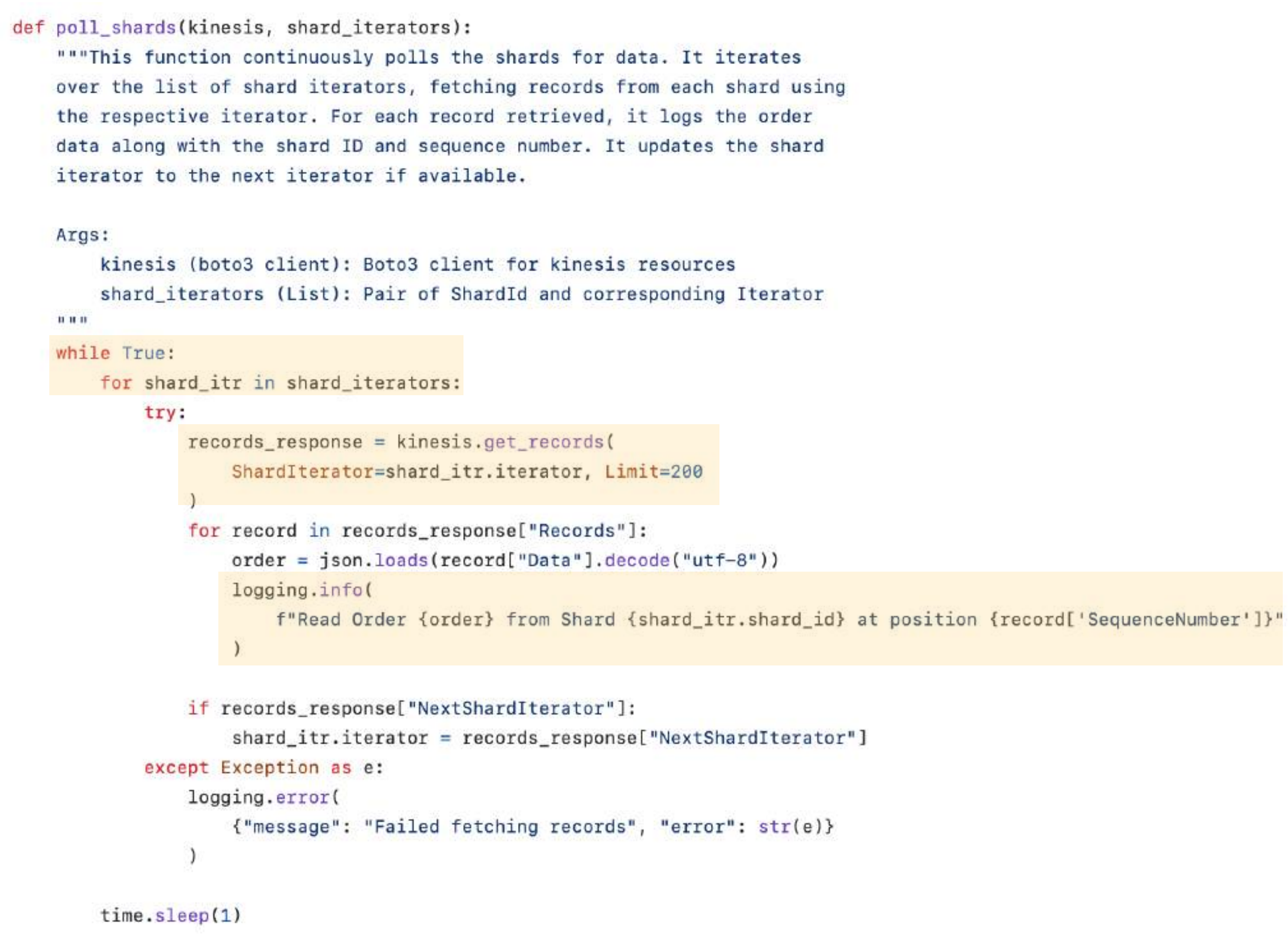
producer_

- writes a s
- uses botc
- can be ru

```
python pr
        --
        --
```

```
eam>
```

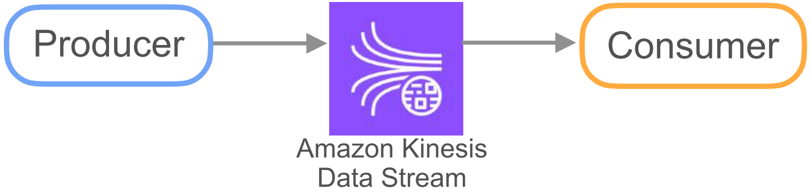
record



```
def poll_shards(kinesis, shard_iterators):
    """This function continuously polls the shards for data. It iterates
    over the list of shard iterators, fetching records from each shard using
    the respective iterator. For each record retrieved, it logs the order
    data along with the shard ID and sequence number. It updates the shard
    iterator to the next iterator if available.

    Args:
        kinesis (boto3 client): Boto3 client for kinesis resources
        shard_iterators (List): Pair of ShardId and corresponding Iterator
    """
    while True:
        for shard_itr in shard_iterators:
            try:
                records_response = kinesis.get_records(
                    ShardIterator=shard_itr.iterator, Limit=200
                )
                for record in records_response["Records"]:
                    order = json.loads(record["Data"].decode("utf-8"))
                    logging.info(
                        f"Read Order {order} from Shard {shard_itr.shard_id} at position {record['SequenceNumber']}"
                    )

                if records_response["NextShardIterator"]:
                    shard_itr.iterator = records_response["NextShardIterator"]
            except Exception as e:
                logging.error(
                    {"message": "Failed fetching records", "error": str(e)}
                )

        time.sleep(1)
```

DeepLe

Part 1



producer_from_cli.py

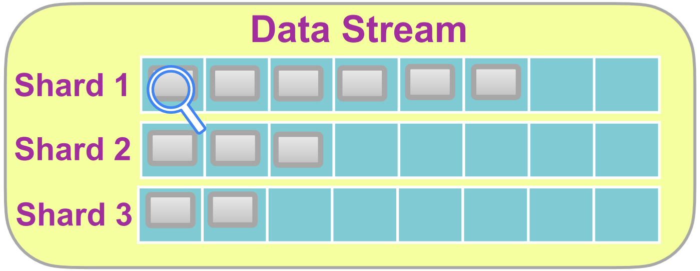
- writes a single data record into the data stream
- uses boto3 to interact with Kinesis
- can be run from the terminal:

```
python producer_from_cli.py
       --stream <name of the data stream>
       --json_string <record as json string>
```

consumer_from_cli.py

- simple consumer application
- uses boto3 to interact with Kinesis
- can be run from the terminal:

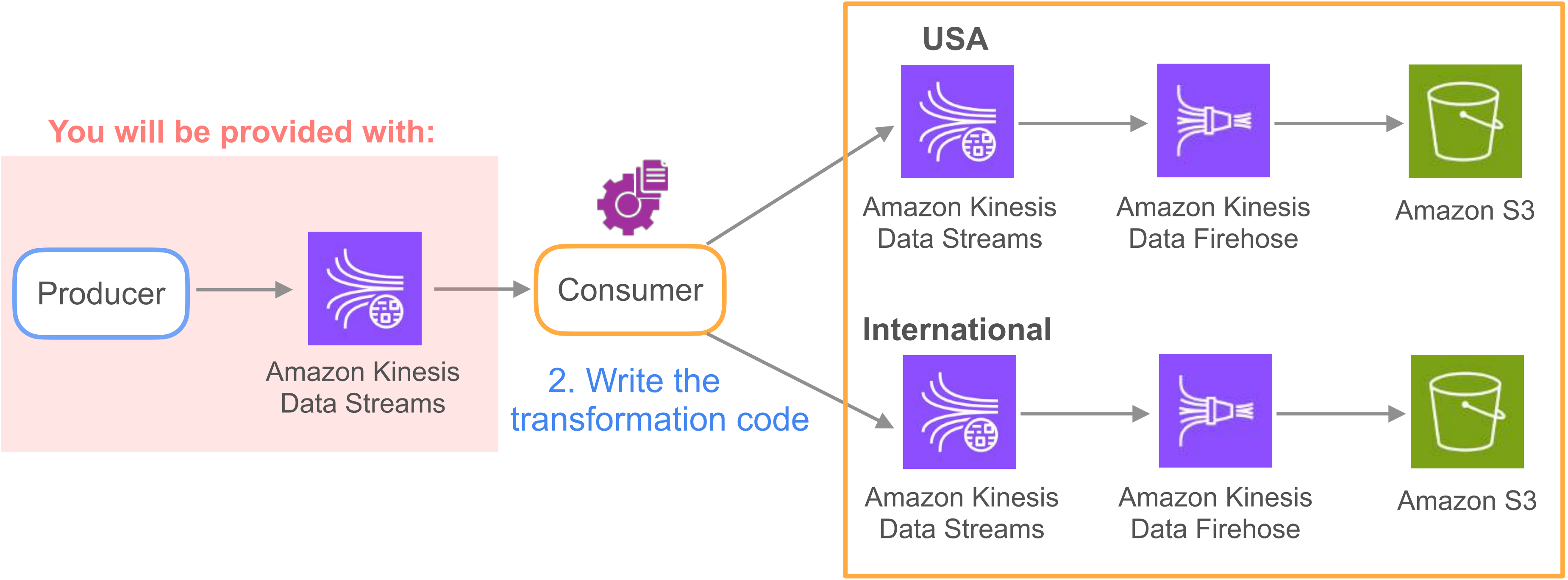
```
python consumer_from_cli.py
       --stream <name of the data stream>
```



print information in the terminal about each record

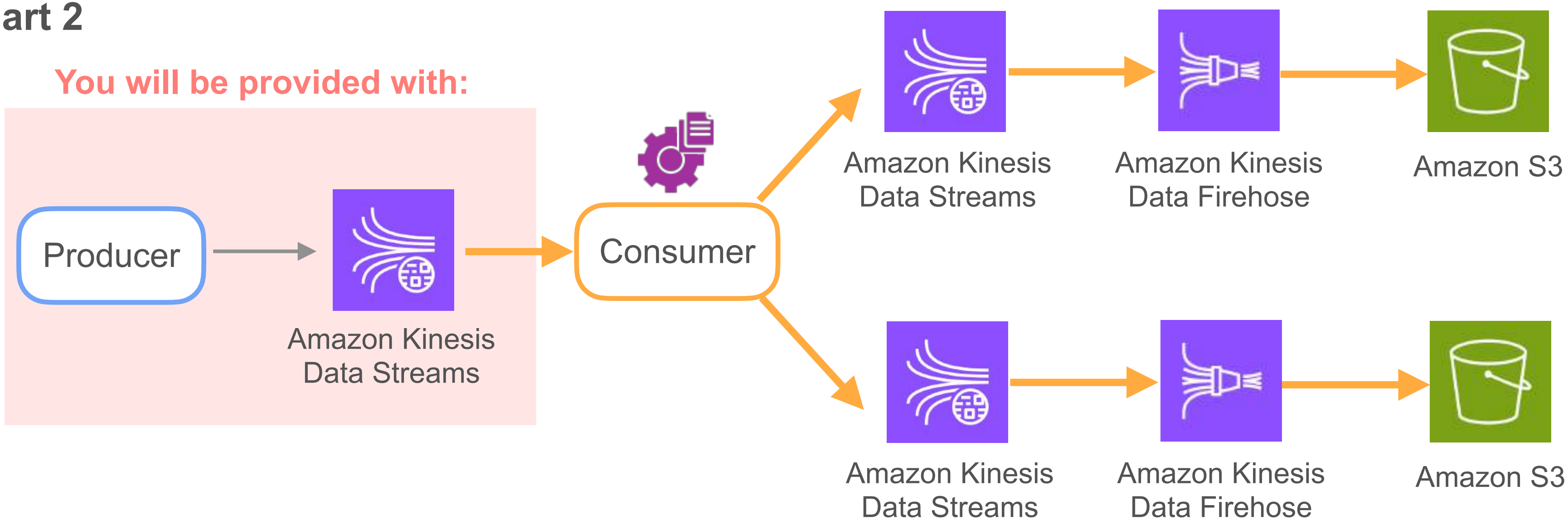
Part 2

Streaming ETL Pipeline

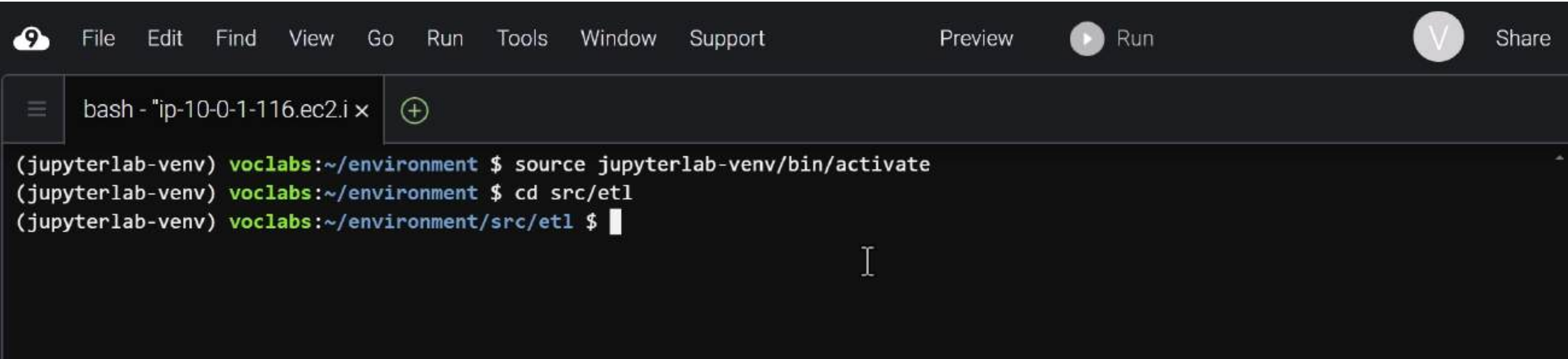

You will be provided with:
Producer
Amazon Kinesis Data Streams
Consumer
2. Write the transformation code
USA
Amazon Kinesis Data Streams
Amazon Kinesis Data Firehose
Amazon S3
International
Amazon Kinesis Data Streams
Amazon Kinesis Data Firehose
Amazon S3

1. Create these resources

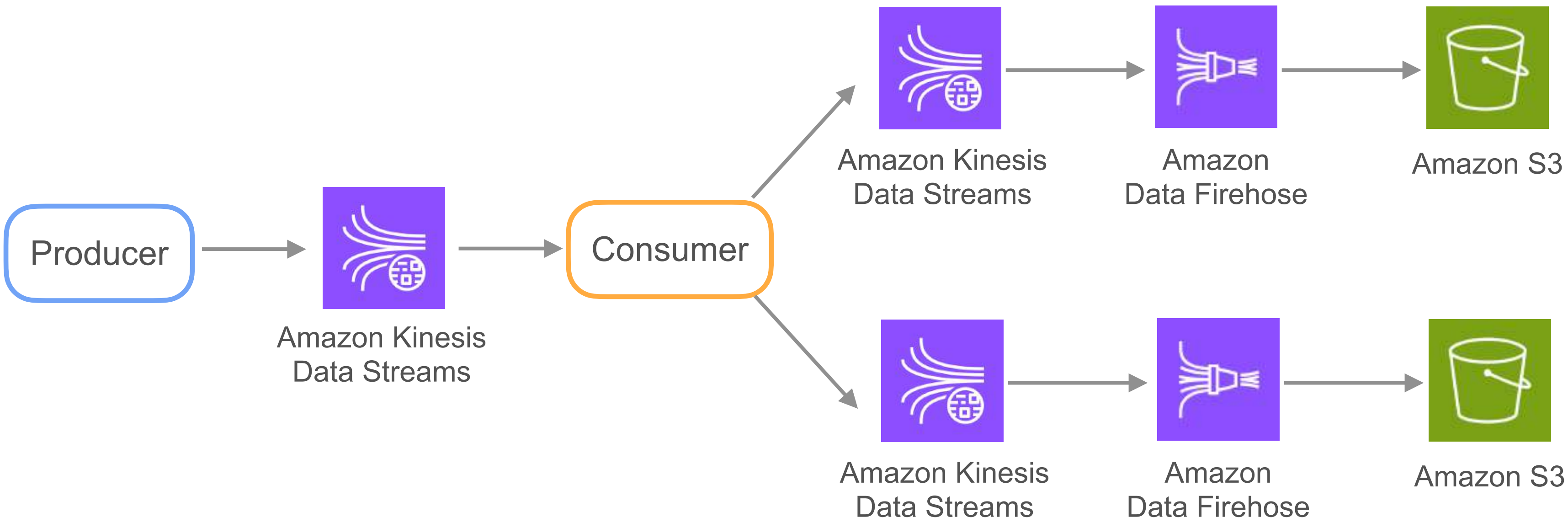
Part 2



You will be provided with:
Producer
Amazon Kinesis Data Streams
Consumer
Amazon Kinesis Data Streams
Amazon Kinesis Data Firehose
Amazon S3
Amazon Kinesis Data Streams
Amazon Kinesis Data Firehose
Amazon S3



```
(jupyterlab-venv) voclabs:~/environment $ source jupyterlab-venv/bin/activate
(jupyterlab-venv) voclabs:~/environment $ cd src/etl
(jupyterlab-venv) voclabs:~/environment/src/etl $ 
```



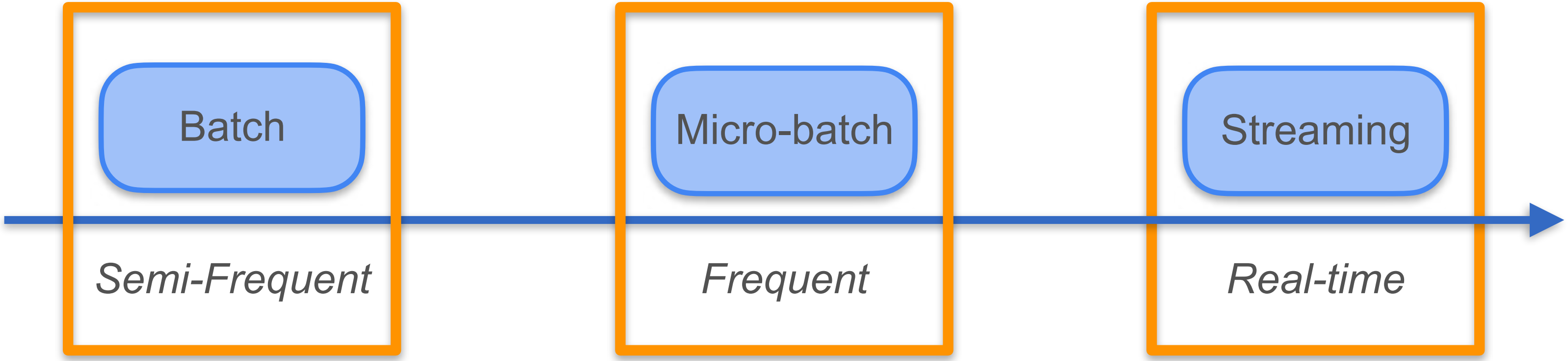
Producer
Amazon Kinesis Data Streams
Consumer
Amazon Kinesis Data Streams
Amazon Data Firehose
Amazon S3
Amazon Kinesis Data Streams
Amazon Data Firehose
Amazon S3

DeepLearning.AI

Data Ingestion

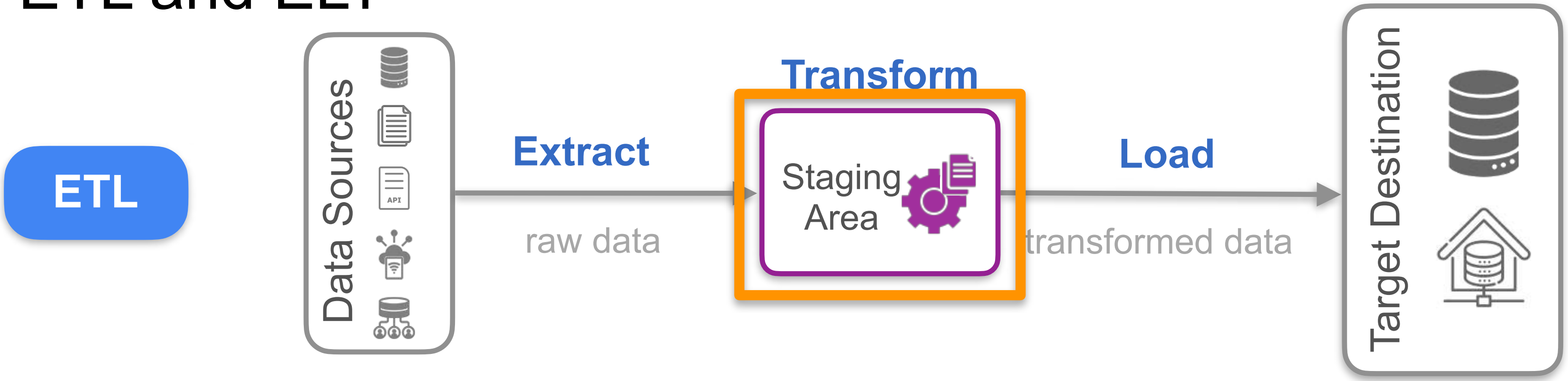
Week 2 Summary

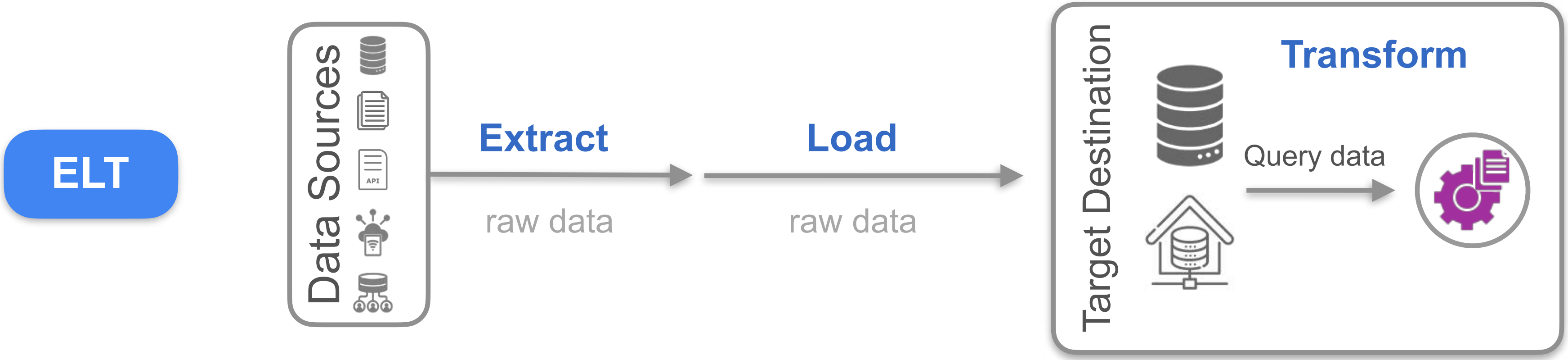
Batch and Streaming Ingestion



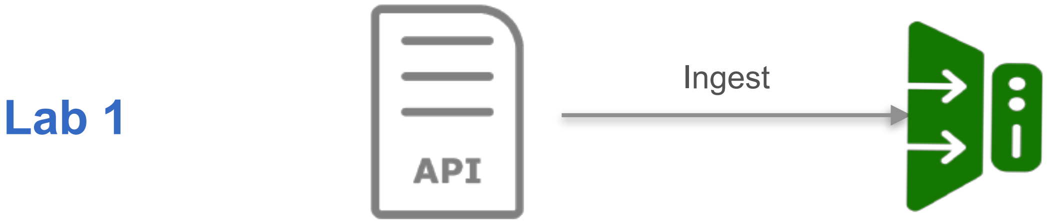
You determine your approach based on the stakeholder needs.

ETL and ELT

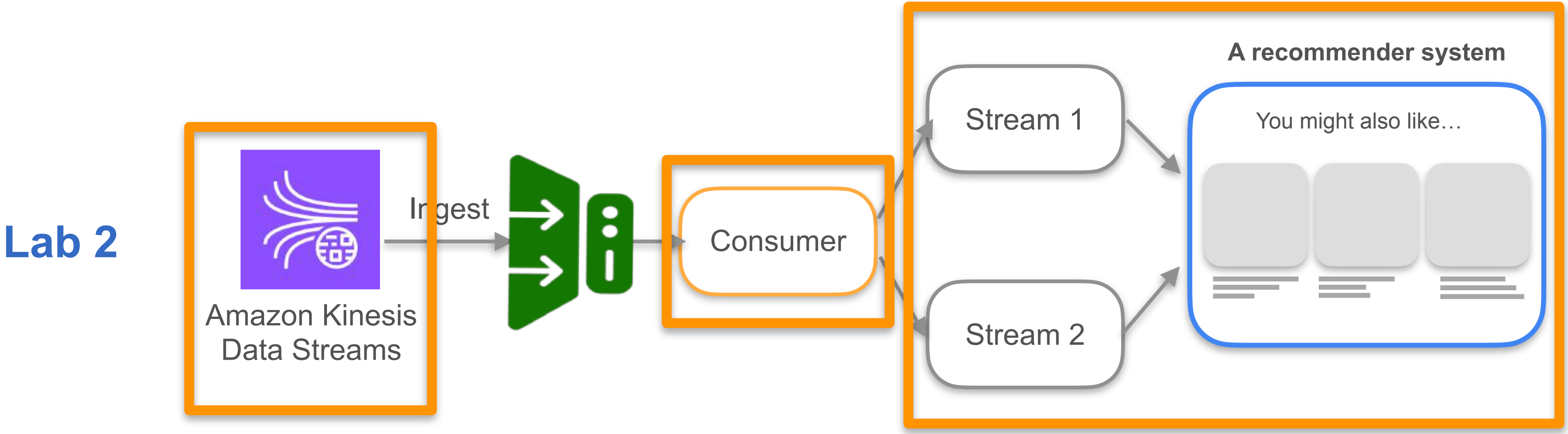

ETL
Data Sources
API
Extract
raw data
Transform
Staging Area
Load
transformed data
Target Destination


ELT
Data Sources
API
Extract
raw data
Load
raw data
Target Destination
Transform
Query data

Week 2 Labs



- Connection to API
- Authentication
- Pagination


Lab 2
Amazon Kinesis Data Streams
Ingest
Consumer
A recommender system
Stream 1
Stream 2
You might also like…