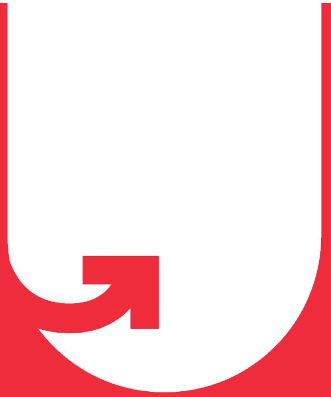
upGrad

STREAM PROCESSING WITH APACHE FLINK

About

upGrad



upGrad

Course: Data Engineering - II

Lecture On: Apache Flink

Instructor: Mayukh Chakraborty

upGrad

INTRODUCTION TO DATASTREAM API

IN THIS SEGMENT

Understand the concept of a DataStream.

Understand the anatomy of a Flink program.

Try a simple Flink program in Java.

DataStream

01	It is a finite, immutable collection of Data objects.
02	It can contain duplicates.
03	Data is read from the source into DataStream. In every transformation step, a new Datastream gets created.

```
DataStream<String> words = …
```

ANATOMY OF A FLINK PROGRAM

1. Obtain an execution environment.
2. Initially load data from the data source.
3. Specify transformations on this data.
4. Specify the data sink.
5. Trigger the program execution.

Validate Email addresses using Apache Flink DataStream

upGrad

STATE AND FAULT TOLERANCE

IN THIS SEGMENT

Understand the concept of State.

Learn about checkpointing and barriers.

Learn about modes of checkpointing.

STATE

01 Stateful functions and operators store data across the processing of individual elements/events.

02 Types of State

03 Embedded database(RocksDb) for storing state data locally.

04 Checkpointing to guarantee persistence of state globally.

FAULT TOLERANCE: CHECKPOINTS

At the heart of Flink's fault tolerance mechanism is drawing snapshots of the distributed data stream and operator state.

The mechanism for drawing these asynchronous snapshots is inspired by the standard Chandy-Lamport algorithm for distributed snapshots.

Savepoints are manually triggered checkpoints and allows updating programs and Flink cluster without losing any state.

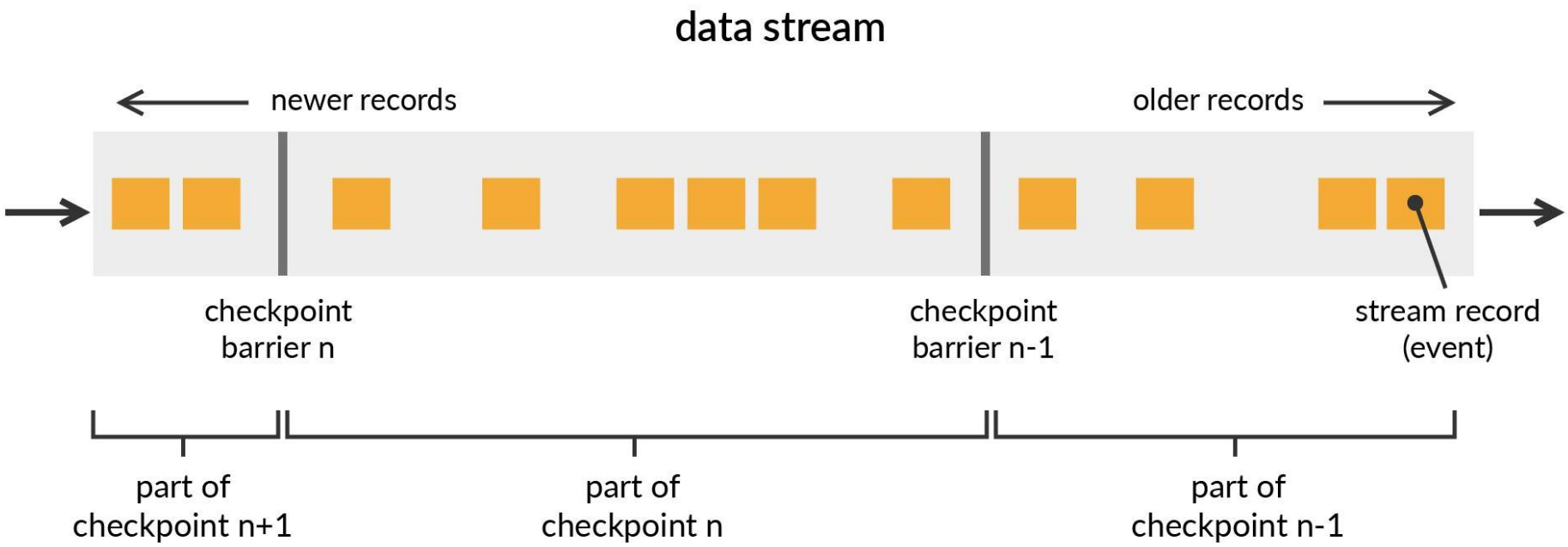
FAULT TOLERANCE: BARRIERS

Distributed snapshotting is based on stream barriers, which are injected into the data stream and flow with the records.

They maintain the sequence and segregates the records that go into the current snapshot, and the records in the next snapshot.

Each barrier carries the ID of the snapshot whose records it pushed in front of it.

Fault Tolerance: Checkpoints & Barriers



upGrad

CHECKPOINTING MODES

CHECKPOINTING MODES

Exactly Once Semantics

1. Flink uses two phase commit protocol for providing exactly once state update.
2. Generally, it adds a small latency of few milliseconds.
3. For applications with super low latency requirement, Flink has a switch to skip the stream alignment during a checkpoint.

At least Once Semantics

1. An operator keeps processing all inputs. The operator also processes elements that belong to checkpoint n+1 before the snapshot for checkpoint n was taken.
2. During restore, these records will result in duplicates, as they are both included in the state snapshot of checkpoint n, and will be replayed as it is.

upGrad

TRANSFORMATIONS

IN THIS SEGMENT

Learn about various types of transformations available with Datastream API.

How transformations can be used in Flink programs?

TRANSFORMATIONS

NAME	USE
Map	Takes element in one format and transform into another format.
FlatMap	Takes one element and produces zero, one, or more elements in same or different format.
Filter	Filter out values, for which the expression returns false value.
Union	Produces the union of two data sets.
Join	Joins two data sets in a manner similar to SQL.

KeyedStream Operators

- **KeyBy Operator**

  Logically partitions a stream into disjoint partitions based on keys.

```
dataStream.keyBy(Transaction::getSourceAccountId)
```

KeyedStream Operators

- **Reduce Operator**

  Combines the current element with the last reduced value and emits the new value.

```
keyedStream.reduce(new ReduceFunction<Integer>() {

        public Integer reduce(Integer value1, Integer value2){

            return value1 + value2;
        }});
```

upGrad

TIME & WINDOWS

IN THIS SEGMENT

Learn about various notions of time..

Learn about the concept of watermarks.

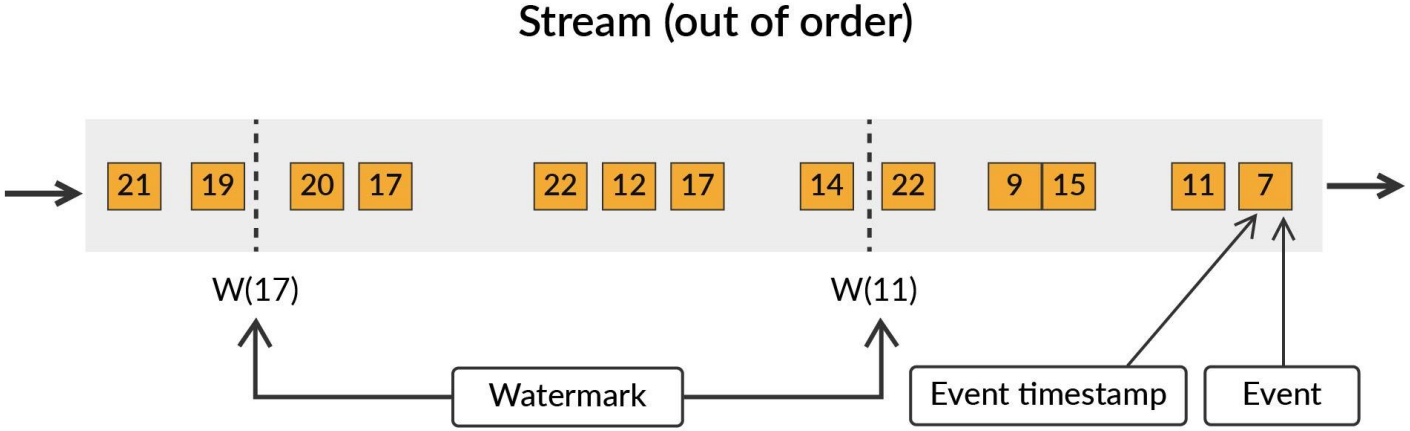
Understand the concept of Window operators.

THE NOTION OF TIME

- **Processing Time-** The clock time of the machine when the respective operation is getting executed.
- **Event Time-** Event time is the time when an event occurred on its producing device. This time is generally embedded within the records before they enter Flink.
- **Ingestion Time-** The time that events enter Flink Environment.

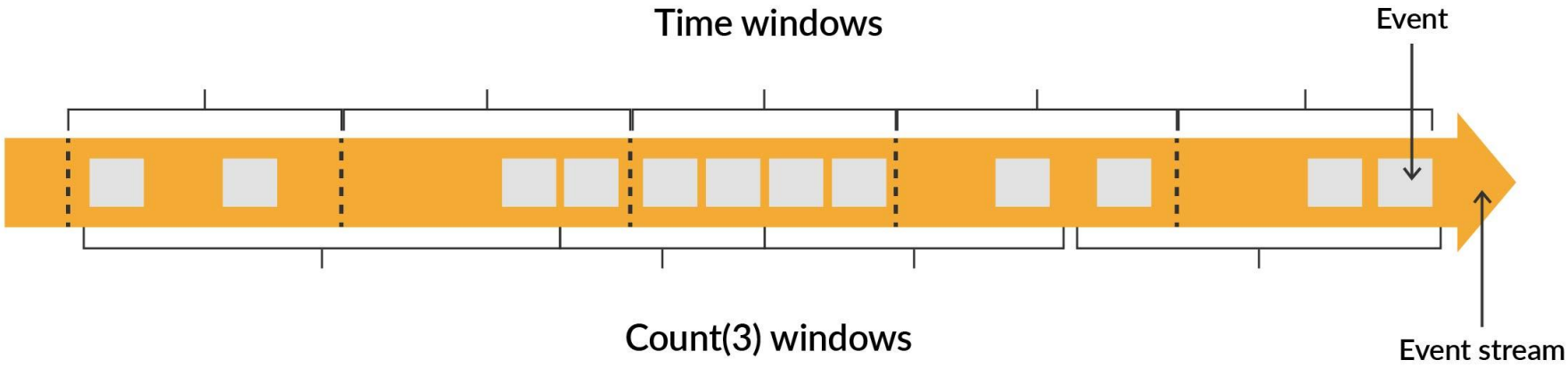
```
env.setStreamTimeCharacteristic(
    TimeCharacteristic.ProcessingTime); // default
```

WATERMARKS



- A Watermark tells operators that no elements with a timestamp older or equal to the watermark timestamp should arrive at the operator.
- Watermarks are emitted at the sources and propagate through the operators. Operators must themselves emit watermarks to downstream operators.

Window Operators



- Aggregation on streams are done on segments of records, called as **windows**.
- Aggregate operations on Windows can be **time driven** (e.g.: count over last 5 minutes) or **data driven** (e.g.: sum of the last 100 elements).

WINDOW OPERATORS

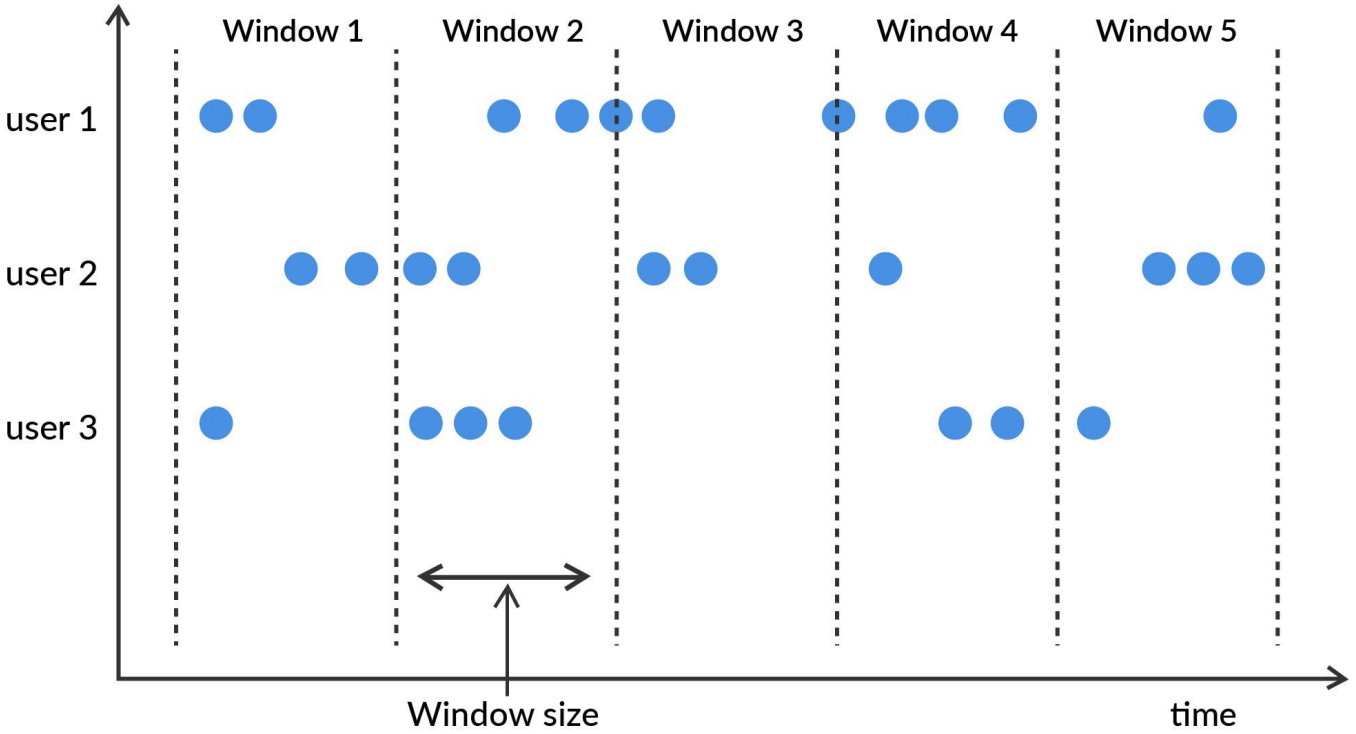
- **Keyed vs Non-Keyed Windows**
  - The first thing to specify is whether your stream should be keyed or not.
  - Using the keyBy(...) will split your infinite stream into logical keyed streams.
  - If keyBy(...) is not called, your stream is not keyed.

WINDOW OPERATORS

- ## Windows Assigner
  - The window assigner defines how elements are assigned to windows. This is done by specifying the WindowAssigner of your choice.
  - Types of Built-in assigners:
    - Tumbling Window Assigner
    - Sliding Window Assigner
    - Session Window Assigner
    - Global Window Assigner

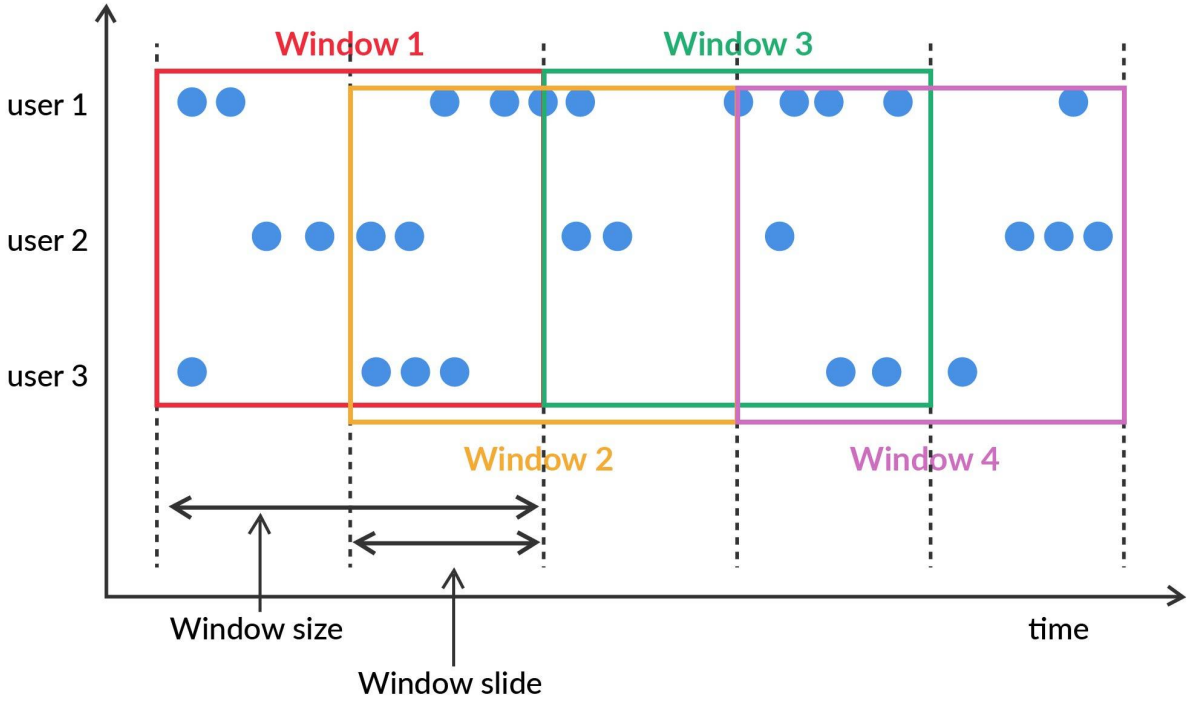
TUMBLING WINDOWS

Assigns each element of a datastream to a fixed window size.



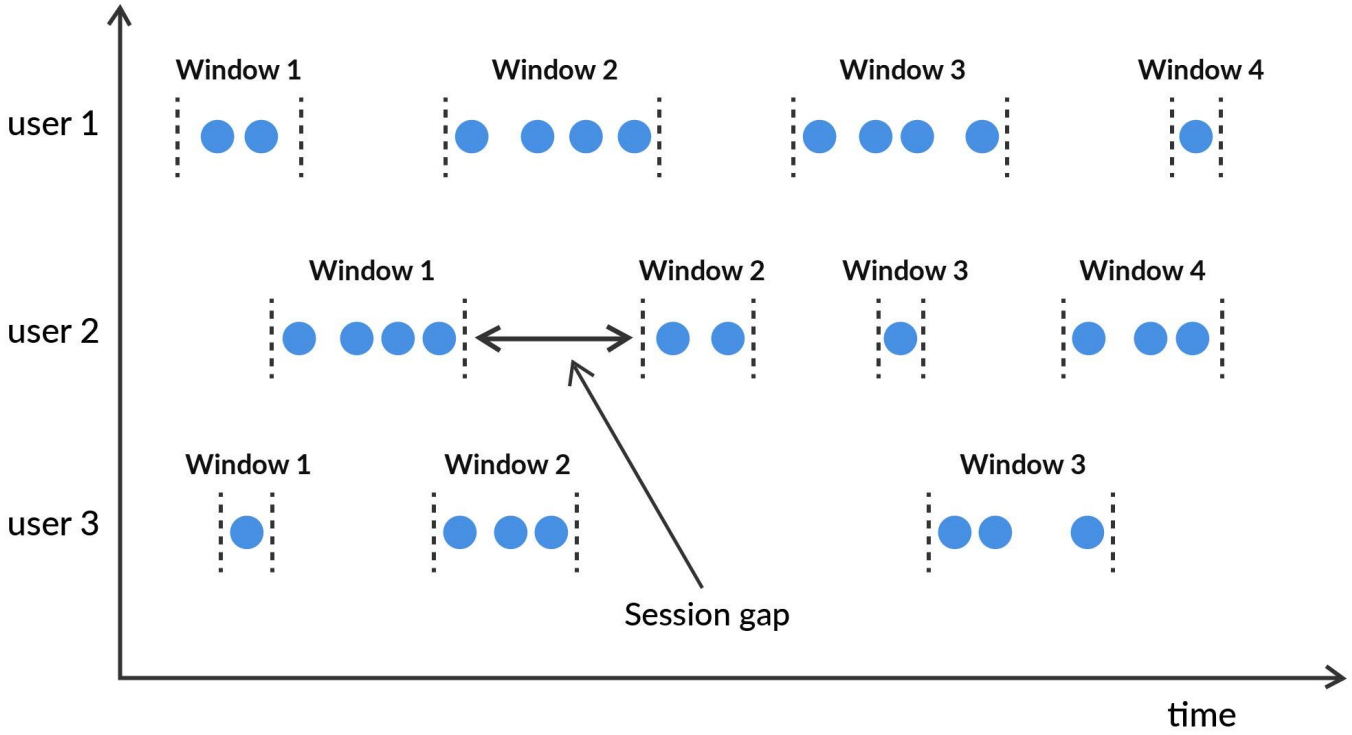
SLIDING WINDOWS

The window size is fixed and there is additional window slide parameter which controls how frequently a sliding window is started.



SESSION WINDOWS

Groups elements by session of activity. Session window does not have a start or an end instead window closes when there is inactivity for a certain period of time.



WINDOW OPERATORS

- **Window Functions**
  - The window functions specify the computation that we want to perform on each of these windows.
  - Types of Built-in functions:
    - Reduce Function
    - Aggregate Function
    - Fold Function
    - Process Function

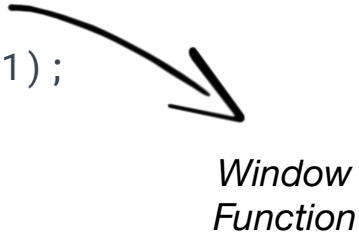
Window Operators

```
DataStream<Tuple2<String, Long>> input = ...;

input
  .keyBy(t -> t.f0)
  ----------------------------------------------

  .window(TumblingEventTimeWindows.of(Time.seconds(5)))
  ----------------------------------------------

  .reduce(new ReduceFunction<Tuple2<String, Long>> {
      public Tuple2<String, Long> reduce(Tuple2<String, Long> v1,
Tuple2<String, Long> v2) {
        return new Tuple2<>(v1.f0, v1.f1 + v2.f1);
      }
    });
```



upGrad

CONNECTORS

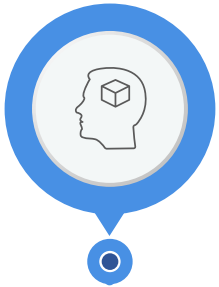
IN THIS SEGMENT

Know about various types of connectors which can be used with Datastream API.

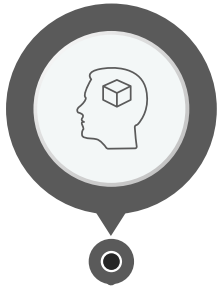
How connectors can be used in Flink programs?

TYPES OF CONNECTORS

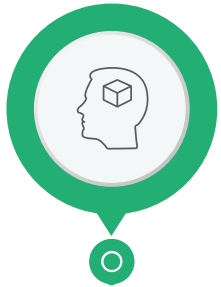
Apache Kafka (Source/Sink)



Apache Cassandra (Sink)



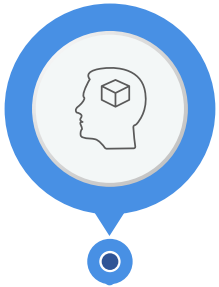
Amazon Kinesis Streams (Source/Sink)



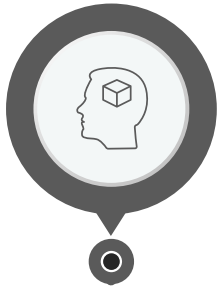
Elastic Search (Sink)

TYPES OF CONNECTORS

Hadoop FileSystem (Sink)



RabbitMQ (Source/Sink)



Google PubSub (Source/Sink)

JDBC (Sink)

upGrad

Exercise