

Implementing Advanced Kinesis Consumers



Ivan Mushketyk

@mushketyk brewing.codes



Overview



Kinesis
Connector
Library

**Library to integrate
Kinesis with other
services**



AWS Lambda

**Serverless computing
platform**

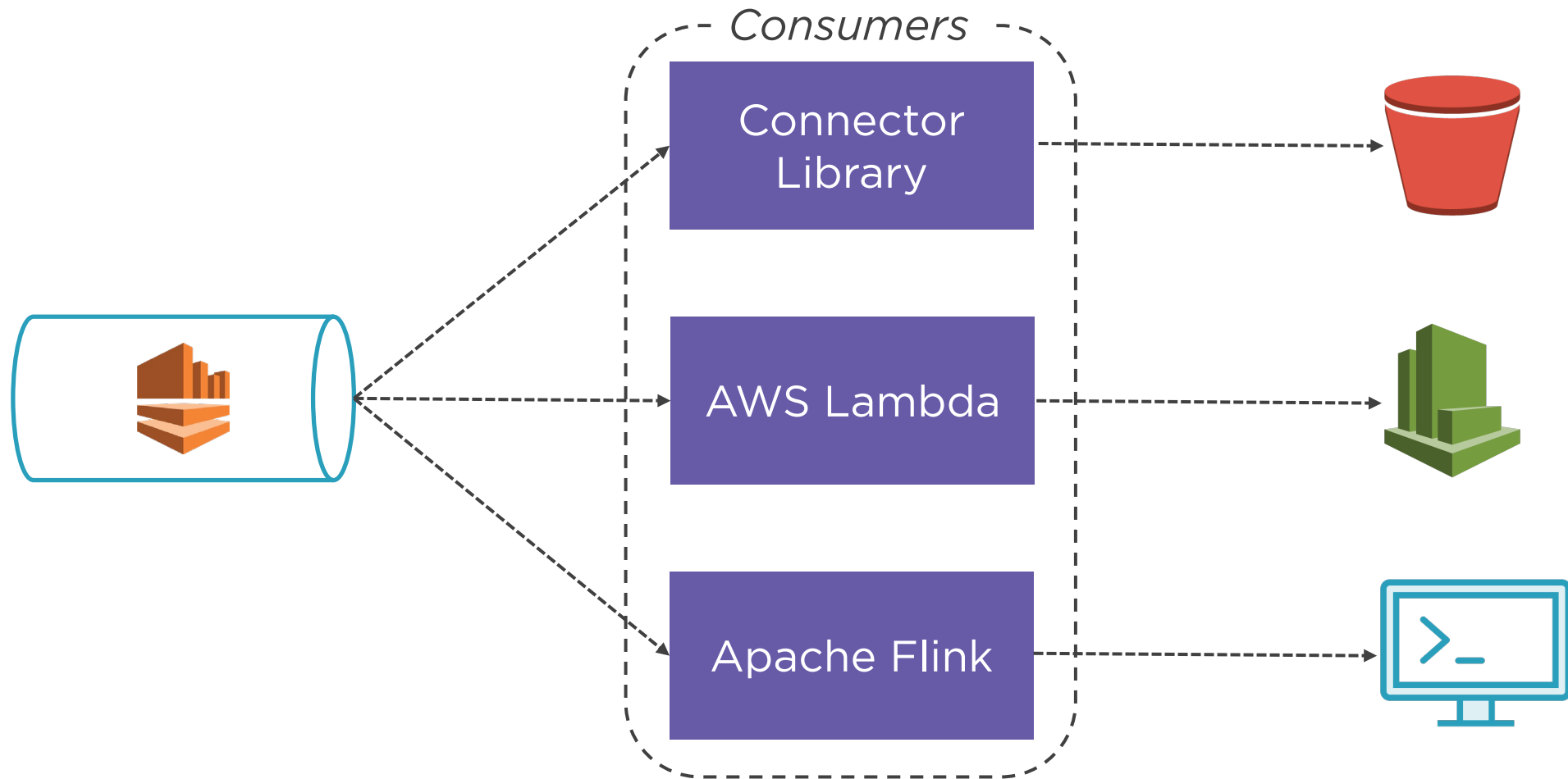


Apache Flink

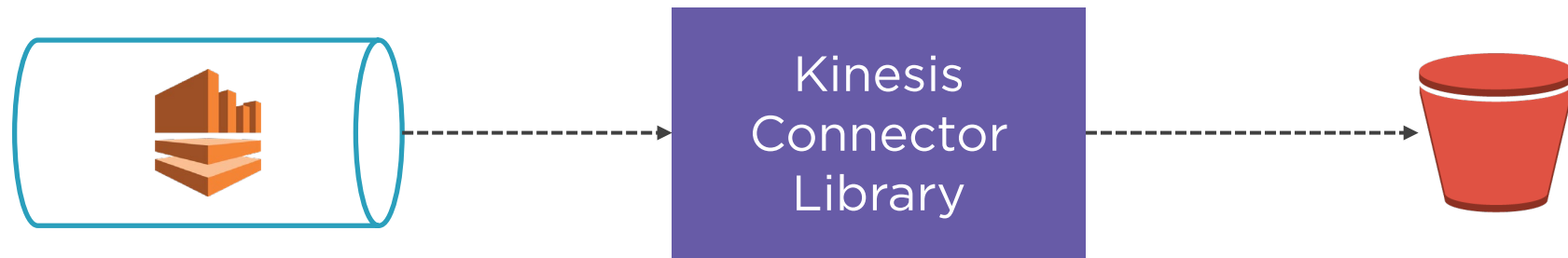
**4G Stream processing
framework
(similar to Apache
Spark)**



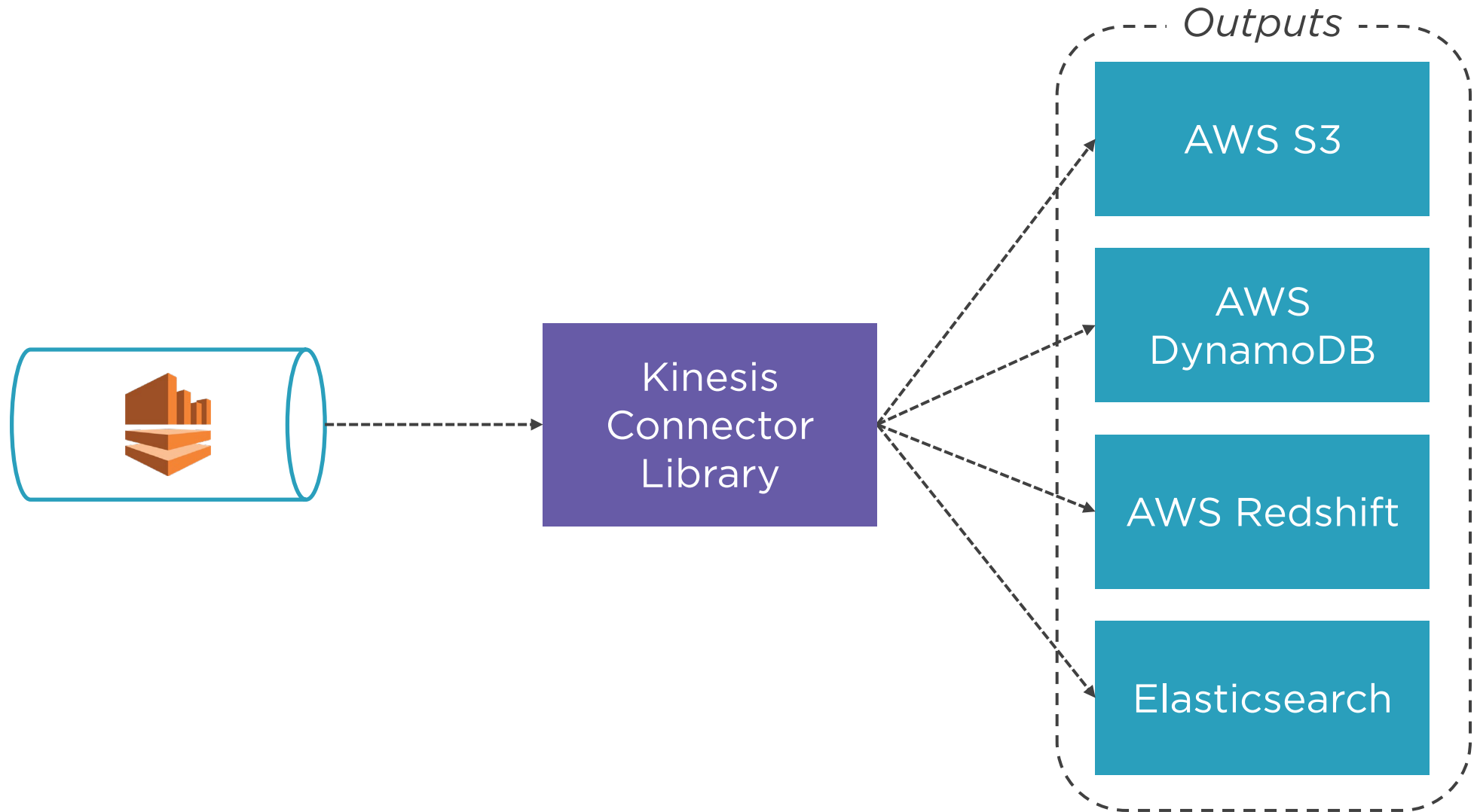
Overview



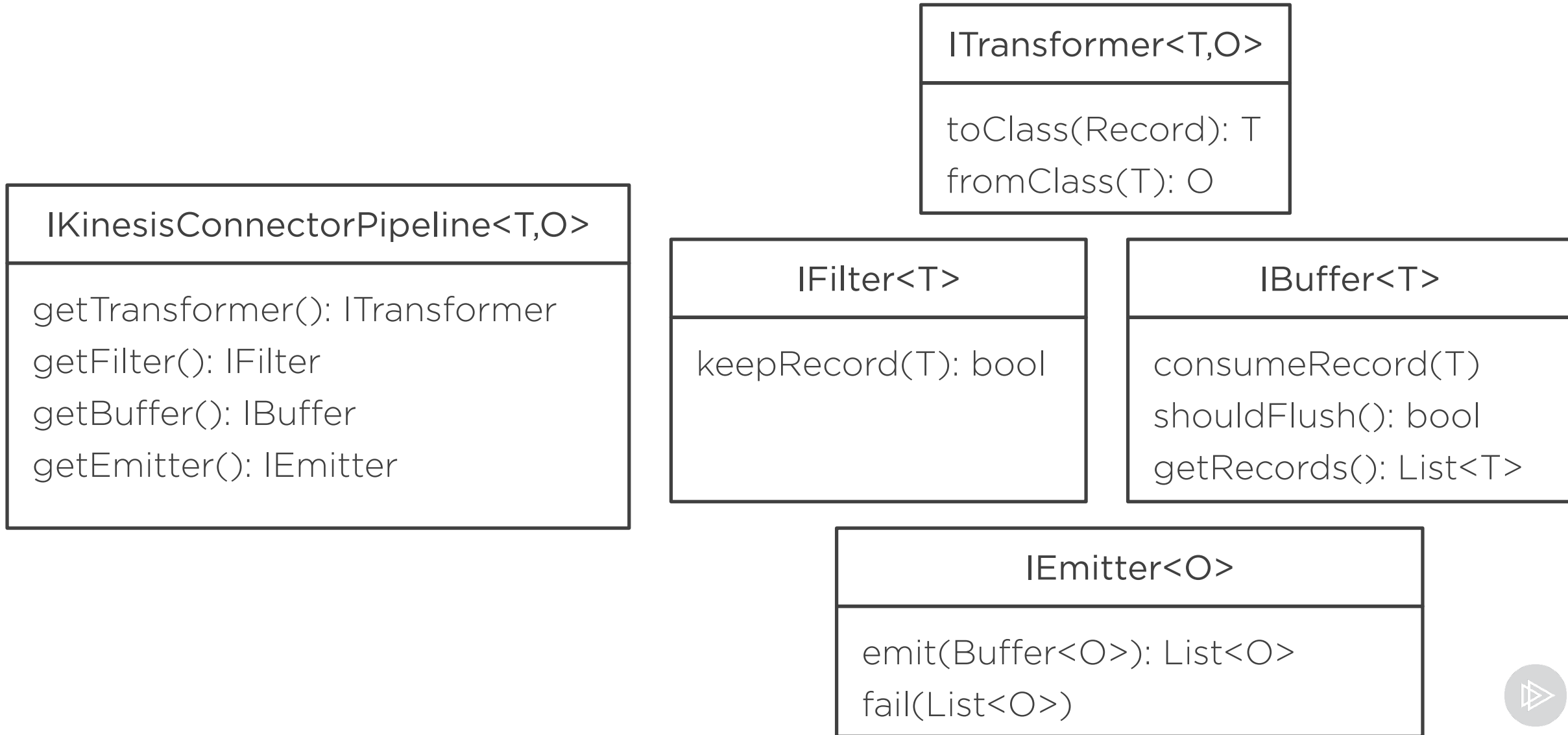
Saving Kinesis Data to S3



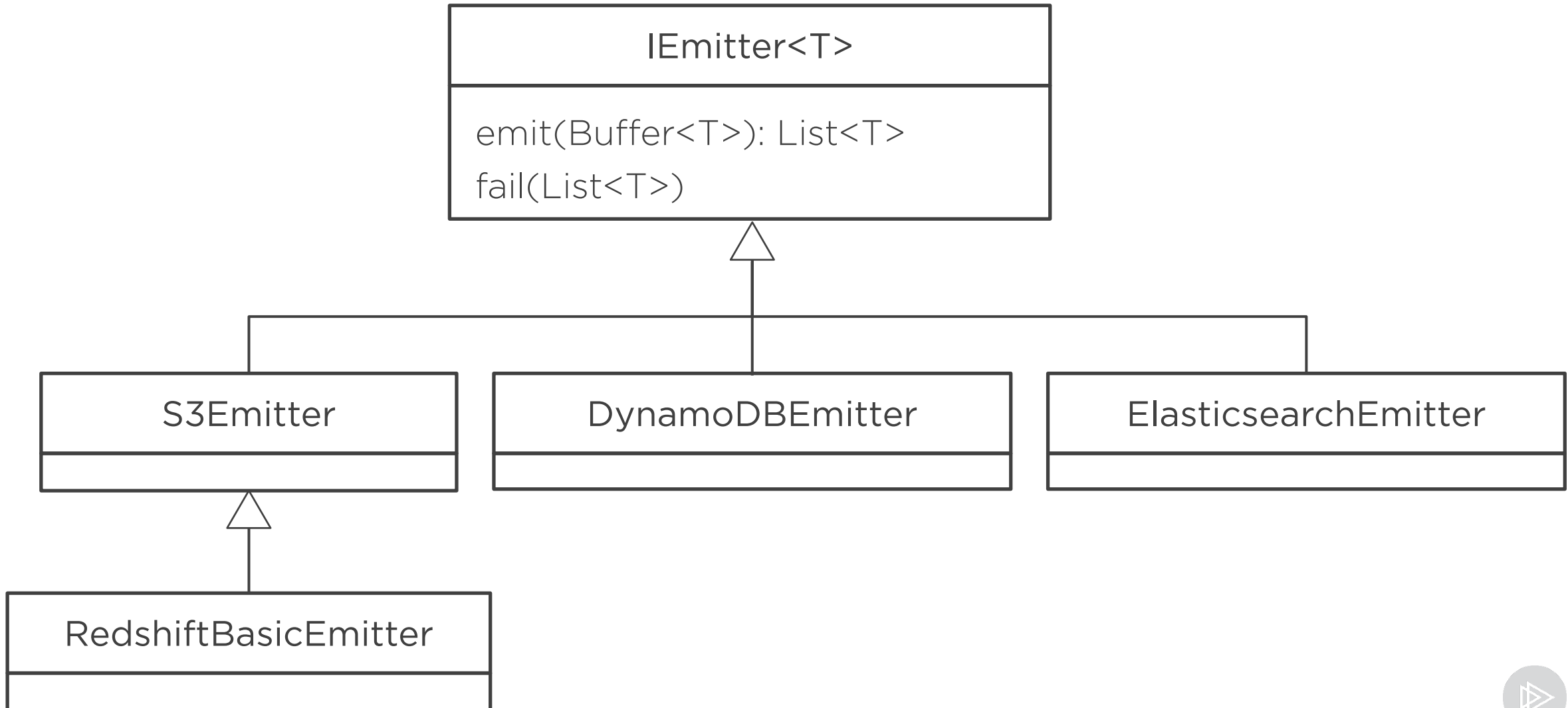
Kinesis Connector Library



Pipeline Stages



Emitters Implementations



Defining a Pipeline

```
public class Pipeline implements
    IKinesisConnectorPipeline<Status, byte[]> {

    @Override
    public ITransformer<Status, byte[]>
    getTransformer(KinesisConnectorConfiguration c) {
        return new StatusTransformer();
    }

    @Override
    public IFilter<Status> getFilter(KinesisConnectorConfiguration c) {
        return new AllPassFilter<KinesisMessageModel>();
    }
}
```



Defining a Pipeline

```
@Override
public IBuffer<Status> getBuffer(KinesisConnectorConfiguration c) {
    return new BasicMemoryBuffer<>(c);
}

@Override
public IEmitter<byte[]> getEmitter(KinesisConnectorConfiguration c) {
    return new S3Emitter(c);
}

}
```



Configuration for Connector Library

```
Properties properties = new Properties();

properties.setProperty(
    KinesisConnectorConfiguration.PROP_APP_NAME,
    "save-to-s3"
);

properties.setProperty(
    KinesisConnectorConfiguration.PROP_S3_BUCKET,
    "pluralsight-kinesis-course"
);

properties.setProperty(
    KinesisConnectorConfiguration.PROP_BUFFER_BYTE_SIZE_LIMIT,
    "100"
);
```



Launching a Pipeline

```
KinesisConnectorConfiguration config = new
KinesisConnectorConfiguration(
    properties,          // Configuration for Connector Library
    new DefaultAWSCredentialsProviderChain()
);

KinesisConnectorRecordProcessorFactory<Status, byte[]> factory =
    new KinesisConnectorRecordProcessorFactory<>(
        new Pipeline(),
        config
    );

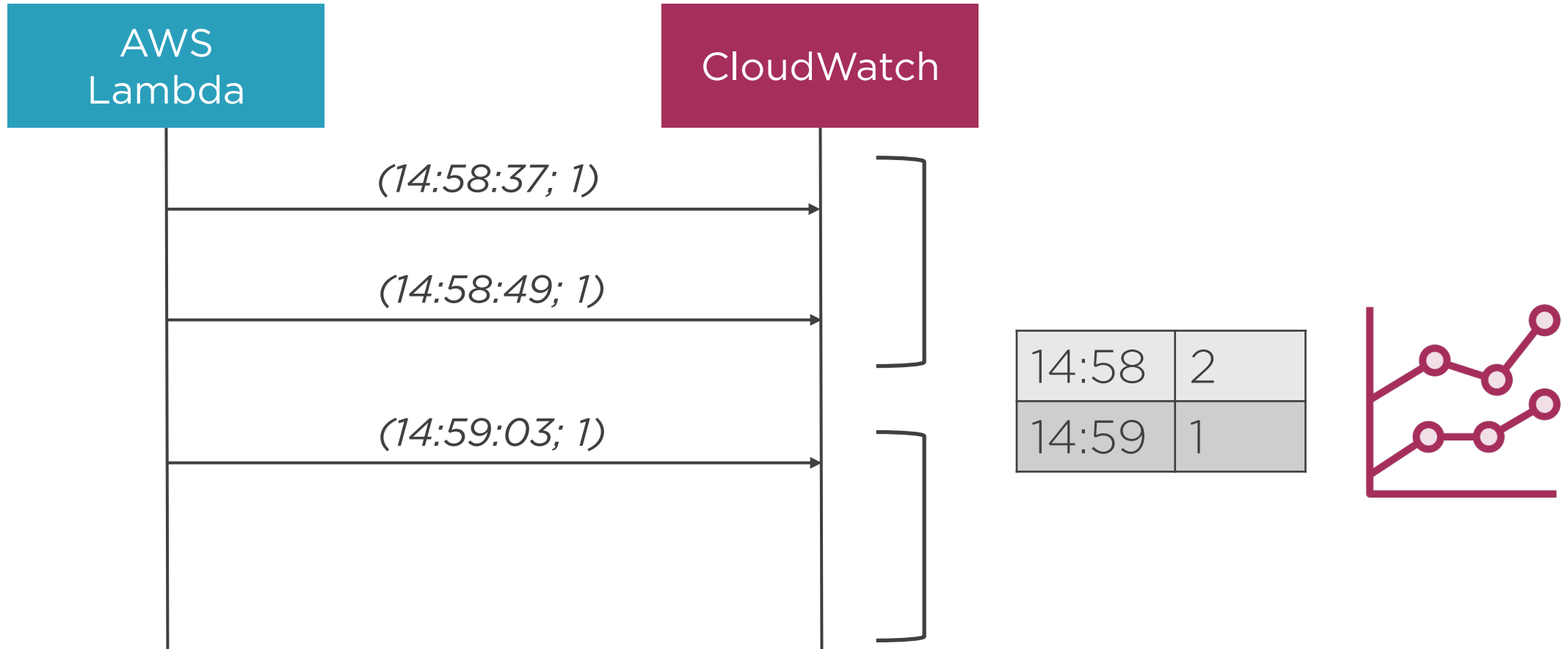
// KCL worker
Worker worker = new Worker.Builder()
    .recordProcessorFactory(factory)
```



Plotting a Number of Tweets



Counting Tweets with CloudWatch



AWS Lambda

Serverless solution

Just need to
upload code

Supports multiple
programming
languages

Integrated with
Kinesis

Automatically
scalable

Cost efficient



Reading Kinesis Data (JavaScript)

```
exports.handler = (event, context, callback) => {  
  event.Records.forEach((record) => {  
    var payload = new Buffer(record.kinesis.data, 'base64')  
                      .toString('ascii');  
    console.log('Decoded payload:', payload);  
  })  
};
```



Processing Aggregated Records

```
var agg = require('aws-kinesis-agg');

event.Records.forEach((record) => {
  agg.deaggregateSync(record.kinesis, true, (err, userRecords) => {
    userRecords.forEach((record) => {
      var tweetData = new Buffer(record.data, 'base64')
        .toString('ascii');
    });
  });
});
```



What Are We Going to Implement?

Tweets

mushketyk 18:55

Working on my #AWS
#Kinesis course

anonymous 18:57

For #Pluralsight?

mushketyk 18:57

#Yep



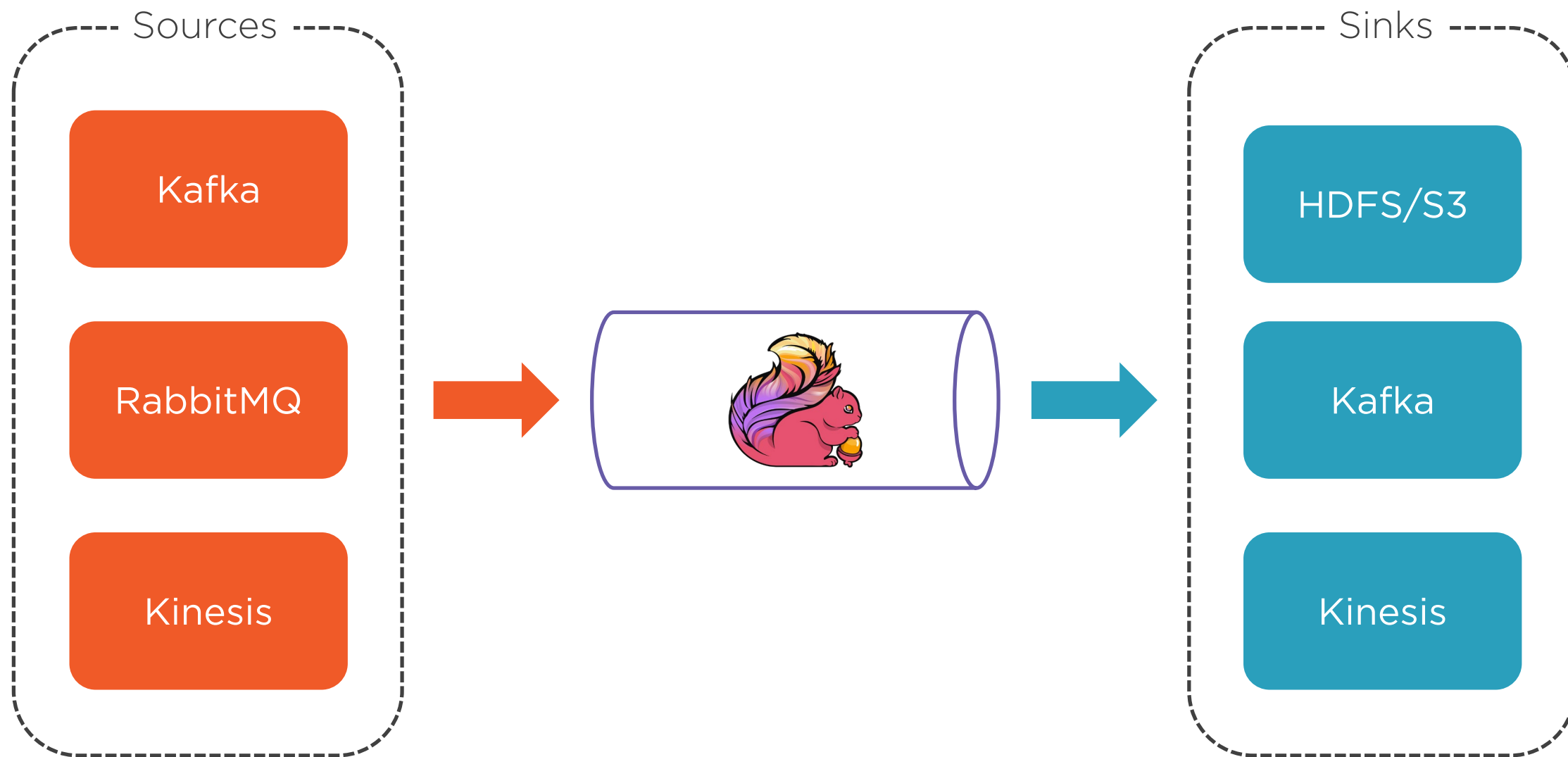
hashtag	time	count
AWS	18:55	1
Kinesis	18:55	1
Pluralsight	18:57	1
Yep	18:57	1



Not a course about Apache Flink. To get more in-depth knowledge, please refer to “Understanding Apache Flink”



Apache Flink



Why Apache Flink?



New 4G streaming
framework



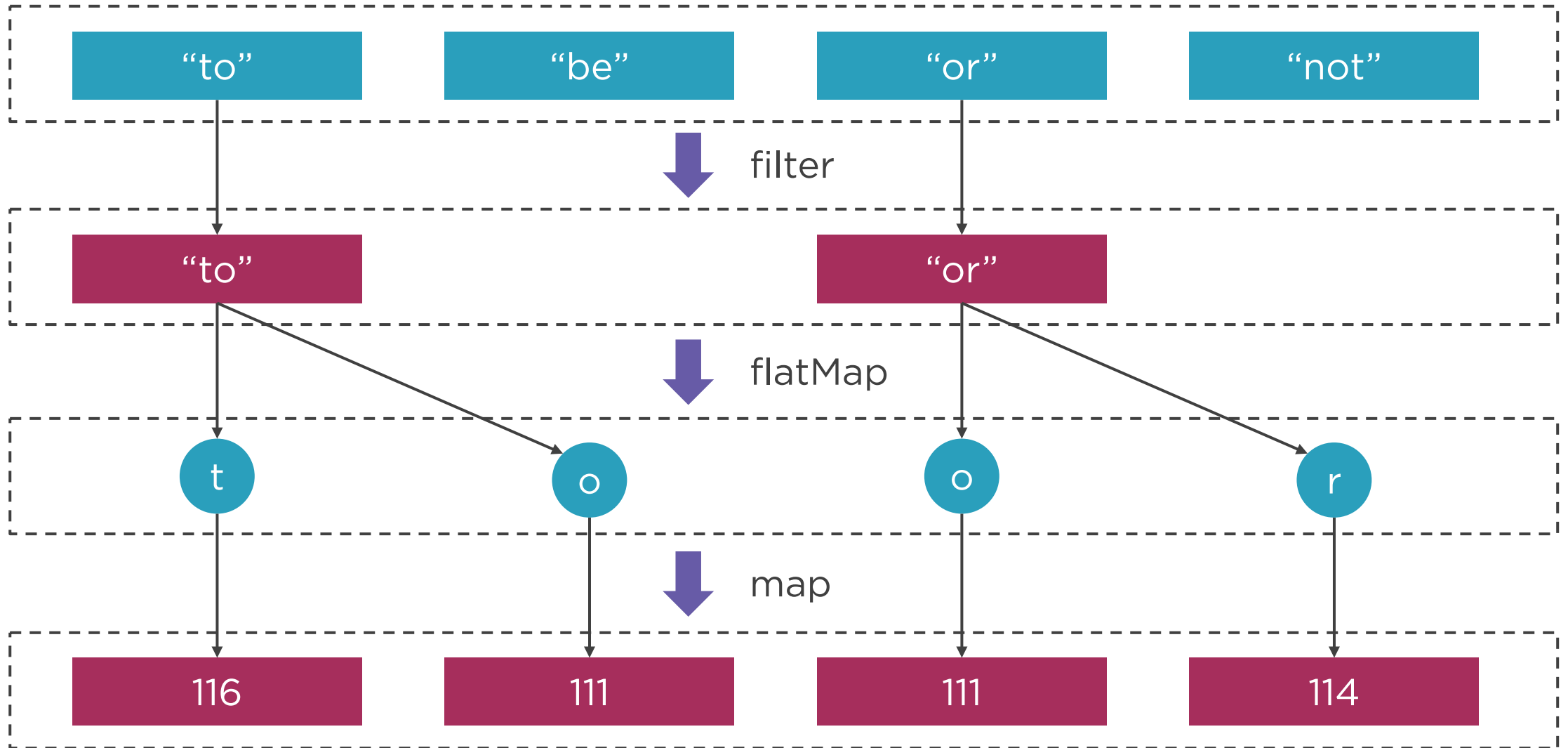
Low latency, high
throughput



Developed ecosystem



Stream Processing with Flink

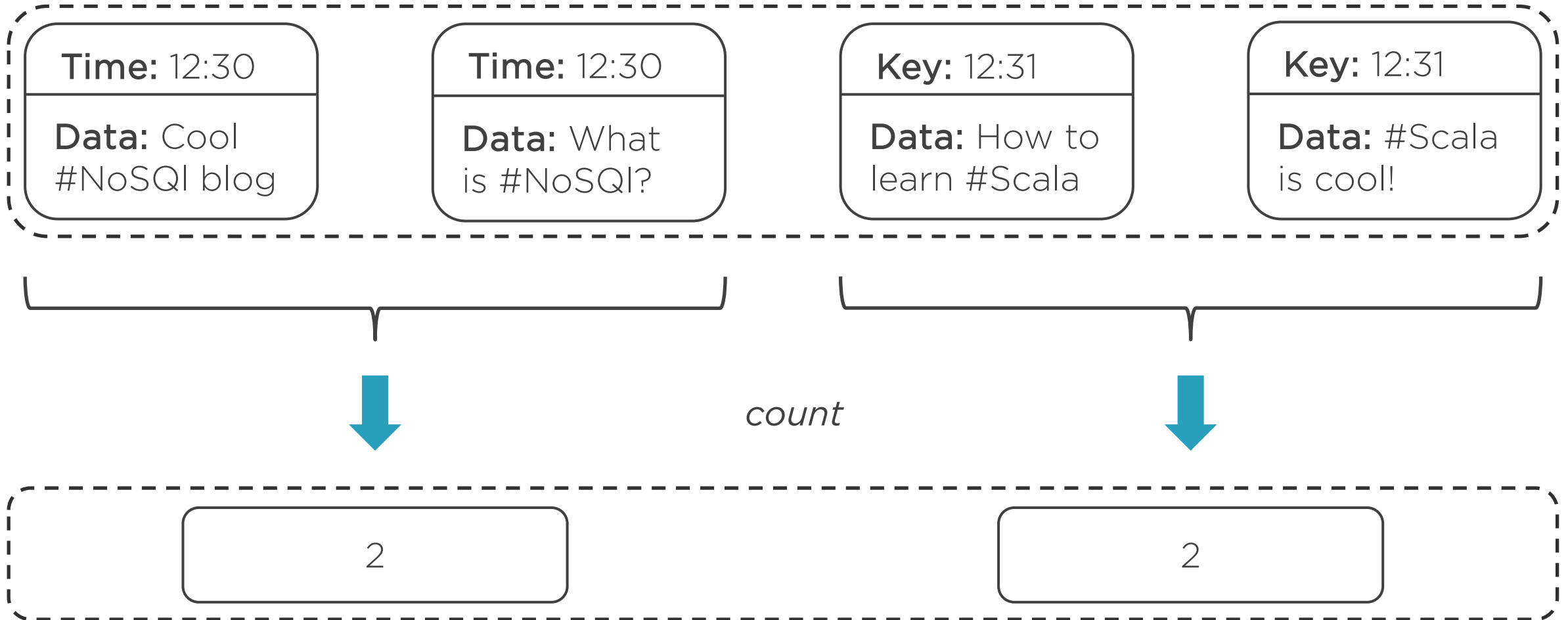


Process Stream Data

```
StreamExecutionEnvironment env =  
StreamExecutionEnvironment.getExecutionEnvironment();  
  
DataStream<Integer> stream = env.fromElements(4, 5, 6, 7, 8, 9);  
  
stream.filter(new FilterFunction<Integer>() {  
    @Override  
    public boolean filter(Integer integer) throws Exception {  
        return integer % 2 == 0;  
    }  
}); // returns DataStream with [4, 6, 8]
```



Processing Windows



Keyed Windows

Hashtags at 11:15

#Flink

#Kinesis

#Pluralsight

#Kinesis



keyBy

#Kinesis

#Kinesis

#Pluralsight

#Flink

Result stream

1

2

1



Create Keyed Window

```
DataStream<String> hashTagStream = ...
hashTagStream
    .keyBy(new KeySelector<String, String>() {
        @Override
        public String getKey(String hashTag) throws Exception {
            return hashTag;
        }
    })
    .timeWindow(Time.minutes(1))
    .apply(...);
```



Hashtag-counting Pipeline

Time window for 12:30

Cool #NoSQL
blog

What is
#NoSQL?

How to learn
#Scala

Cool #NoSQL
blog

flatMap

NoSQL

NoSQL

Scala

Scala

keyBy

Scala

Scala

NoSQL

NoSQL

12:30, Scala, 2

12:30, NoSQL, 2



Reading Data from Kinesis

```
Properties consumerConfig = new Properties();  
consumerConfig.put(ConsumerConfigConstants.AWS_REGION,  
    "us-east-1");  
consumerConfig.put(ConsumerConfigConstants.AWS_ACCESS_KEY_ID,  
    "...");  
consumerConfig.put(ConsumerConfigConstants.AWS_SECRET_ACCESS_KEY,  
    "...");  
consumerConfig.put(ConsumerConfigConstants.STREAM_INITIAL_POSITION,  
    "TRIM_HORIZON");
```



Reading Data from Kinesis

```
StreamExecutionEnvironment env =  
StreamExecutionEnvironment.getExecutionEnvironment();  
  
// Create Kinesis stream  
DataStream<Record> kinesis = env.addSource(  
    new FlinkKinesisConsumer<>(  
        "tweets-stream",  
        new RecordKinesisDeserializationSchema(),  
        consumerConfig)  
);
```



Summary

