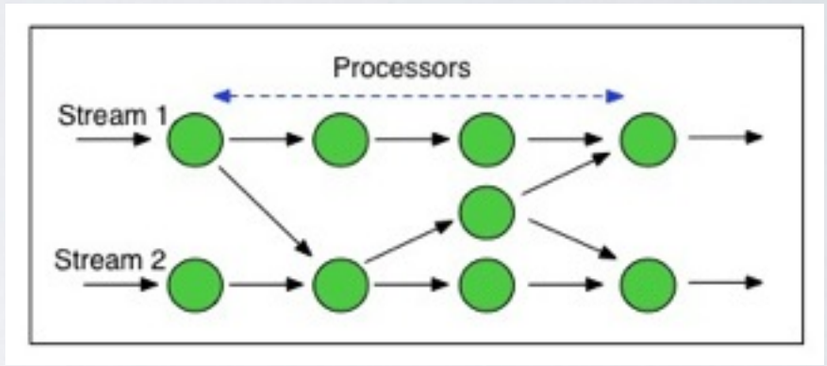
KINESIS @ TOTANGO

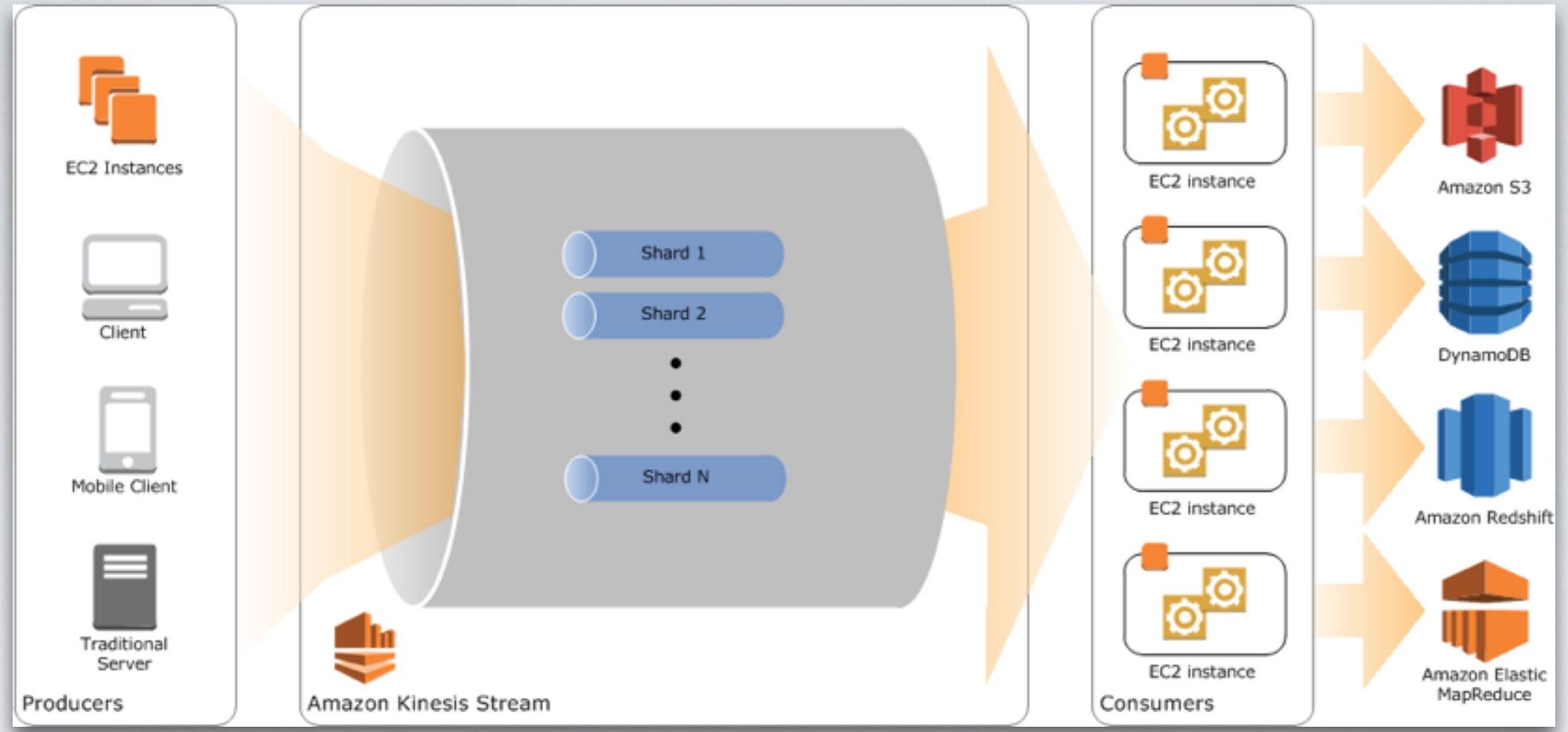
Ran Tavory

WHAT IS KINESIS

- Stream processing - as a service
  - Similar to Kafka



- Use cases:
  - Realtime data stream processing
  - Aggregations
  - Moving time-window, etc..



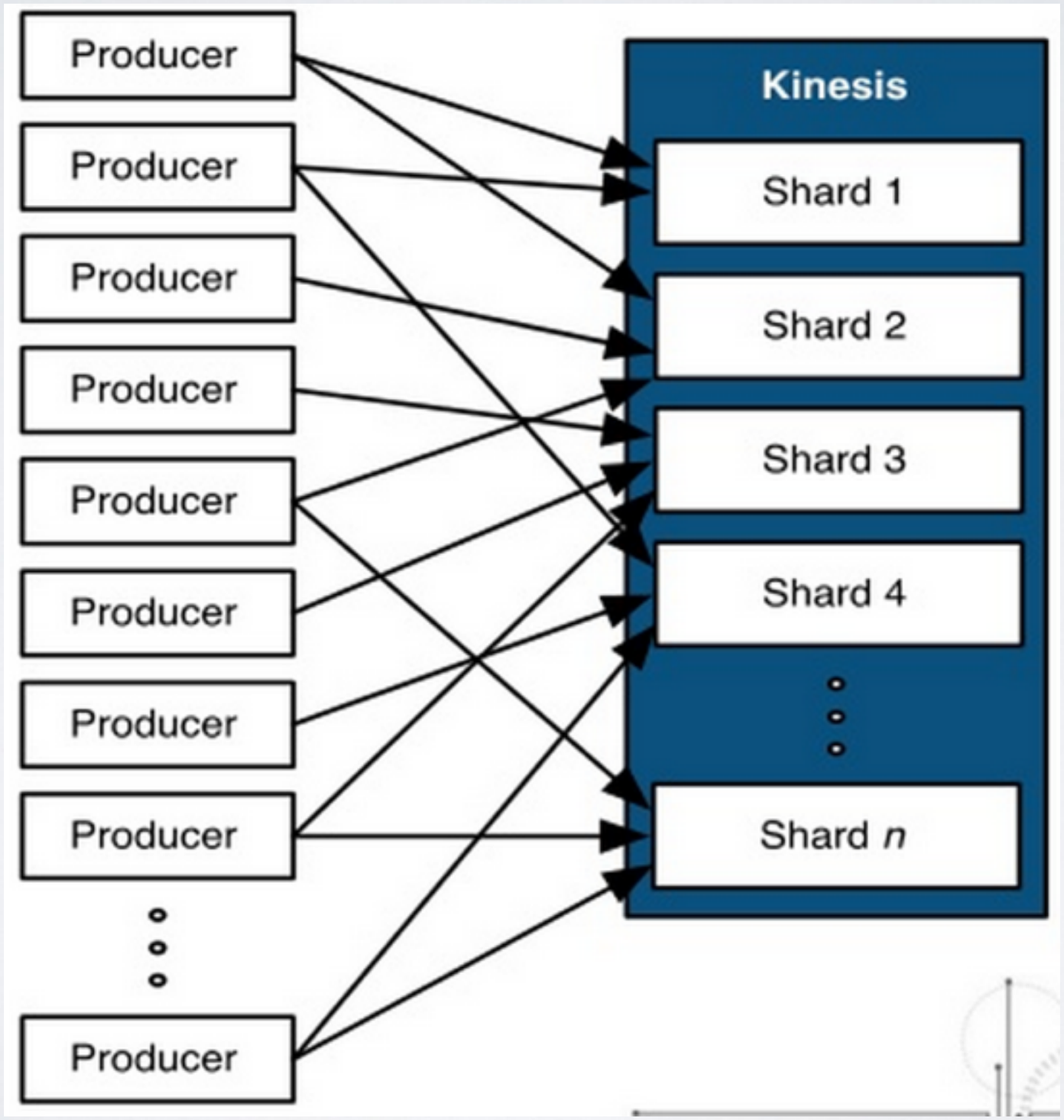
HIGH LEVEL

Overview

CONCEPTS

- Data Record
- Stream
- Partition Key
- Shard
- Sequence Number
- Worker

WRITING

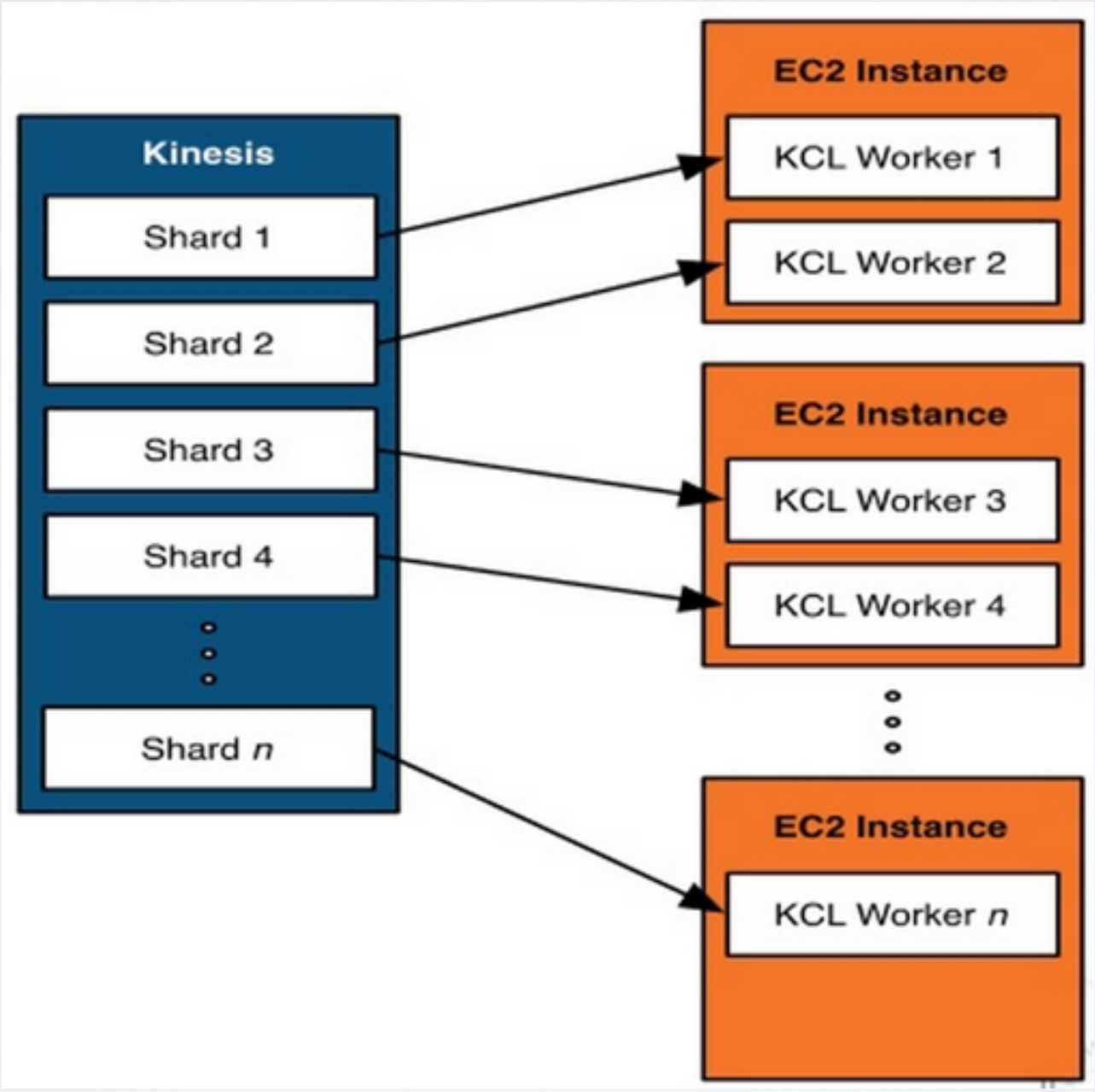

Producer
Producer
Producer
Producer
Producer
Producer
Producer
Producer
Kinesis
Shard 1
Shard 2
Shard 3
Shard 4
Shard n

WRITING

```
PutRecordResult putRecordResult = client.putRecord(putRecordRequest)

// OR:

client.putRecordAsync(putRecordRequest, asyncHandler)
```

```
// In general:
PutRecord(Data, PartitionKey, StreamName)
```

READING


Kinesis
Shard 1
Shard 2
Shard 3
Shard 4
Shard n
EC2 Instance
KCL Worker 1
KCL Worker 2
EC2 Instance
KCL Worker 3
KCL Worker 4
EC2 Instance
KCL Worker n

READING

- Ordered
- Sequence Numbers
- Checkpoints
- “At least once” semantics
- Replay-ability (AT or AFTER _SEQUENCE_NUMBER)
- Iterator types:
    - AT_SEQUENCE_NUMBER, AFTER_SEQUENCE_NUMBER, TRIM_HORIZON (first avail), LATEST (most fresh)

KCL (READING)

- Kinesis Client Library
- Java: https://github.com/awslabs/amazon-kinesis-client
- (there are also for node, python, ruby)

KCL (READING)

- **Connects** to the stream
- **Enumerates** the shards
- **Coordinates** shard associations with other workers (if any)
- Instantiates a **record processor** for every shard it manages
- **Pulls data** records from the stream
- **Checkpoints** processed records
- **Balances** shard-worker associations when the worker instance count changes
- **Balances** shard-worker associations when shards are split or merged

KCL (JAVA)

```
// Processor (worker)
public interface IRecordProcessor {

   void initialize(String shardId);

   void processRecords(List<Record> records, IRecordProcessorCheckpointer checkpointer);

   void shutdown(IRecordProcessorCheckpointer checkpointer, ShutdownReason reason);
}

// Factory
public interface IRecordProcessorFactory {

   IRecordProcessor createProcessor();

}
```

RE-SHARDING

- So what's the story with re-sharding?
  - Choosing the partition key
- Shard limits
  - 1Mb/S ingest, 2Mb/S egress, 1K inserts/S
- Resharding - It is painful :-(

RE-SHARDING (EXAMPLE)

```
$ aws kinesis describe-stream --stream-name
  gateway-received
```

```
$ aws kinesis split-shard --stream-name
  gateway-received --shard-to-split
  shardId-000000000017 --new-starting-hash-key
  255211775190703847597530955573826158591
```

KINESIS V/S SQS

- Speed
- Data item size
- Ordered messages
- Replay-ability
- Sharding (persistent routing)
- Kinesis workers - v/s SQS consumers

KINESIS CONNECTORS

- https://github.com/awslabs/amazon-kinesis-connectors
- Amazon DynamoDB
- Amazon Redshift
- Amazon S3
- Elasticsearch

PRICING

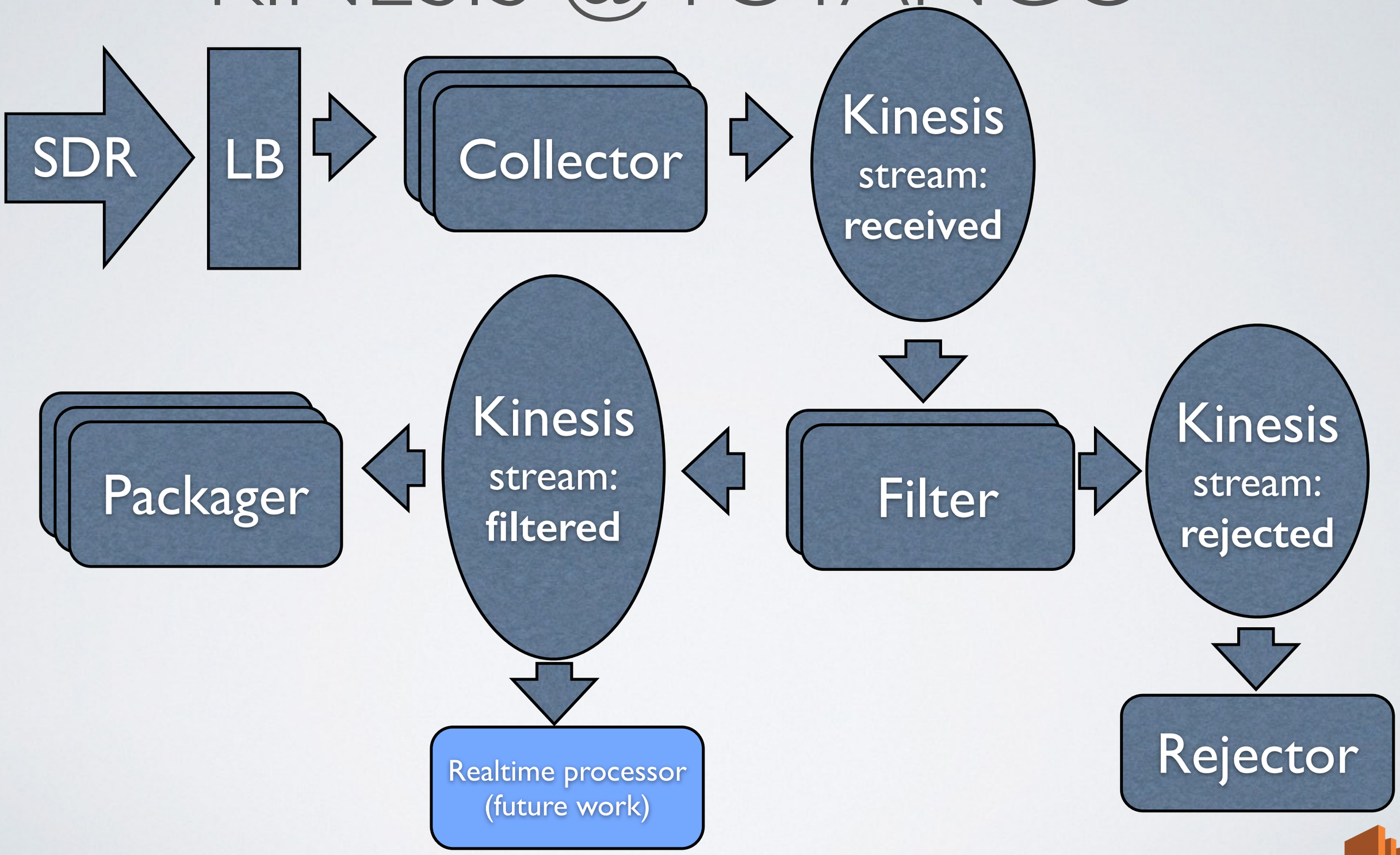
Pricing Dimension	Value
Hourly Shard Rate	$0.015
Per 1,000,000 PUT transactions:	$0.028

- Shard / month => 11$

TOTANGO CLASSES

```
/**
 * A high-level client for Kinesis
 * @author ran
 */
public class KinesisClient {

   public void connect() {

   public List<String> listStreams() {

   public boolean isStreamExists(final String streamName)

   public String describeStream(final String streamName)

   public List<Shard> describeStreamShards(…)

   public List<Shard> describeStreamLeafShards(…)

   public void createStream(…)

   public String putRecord(…)

   public void putRecordAsync(…)
}
```

KINESIS @ TOTANGO



REFERENCES

- http://aws.amazon.com/kinesis/
- http://docs.aws.amazon.com/kinesis/latest/dev/developing-consumer-apps-with-kcl.html
- https://github.com/awslabs/amazon-kinesis-client
- https://github.com/awslabs/amazon-kinesis-connectors
- This presentation: https://speakerdeck.com/rantav/kinesis