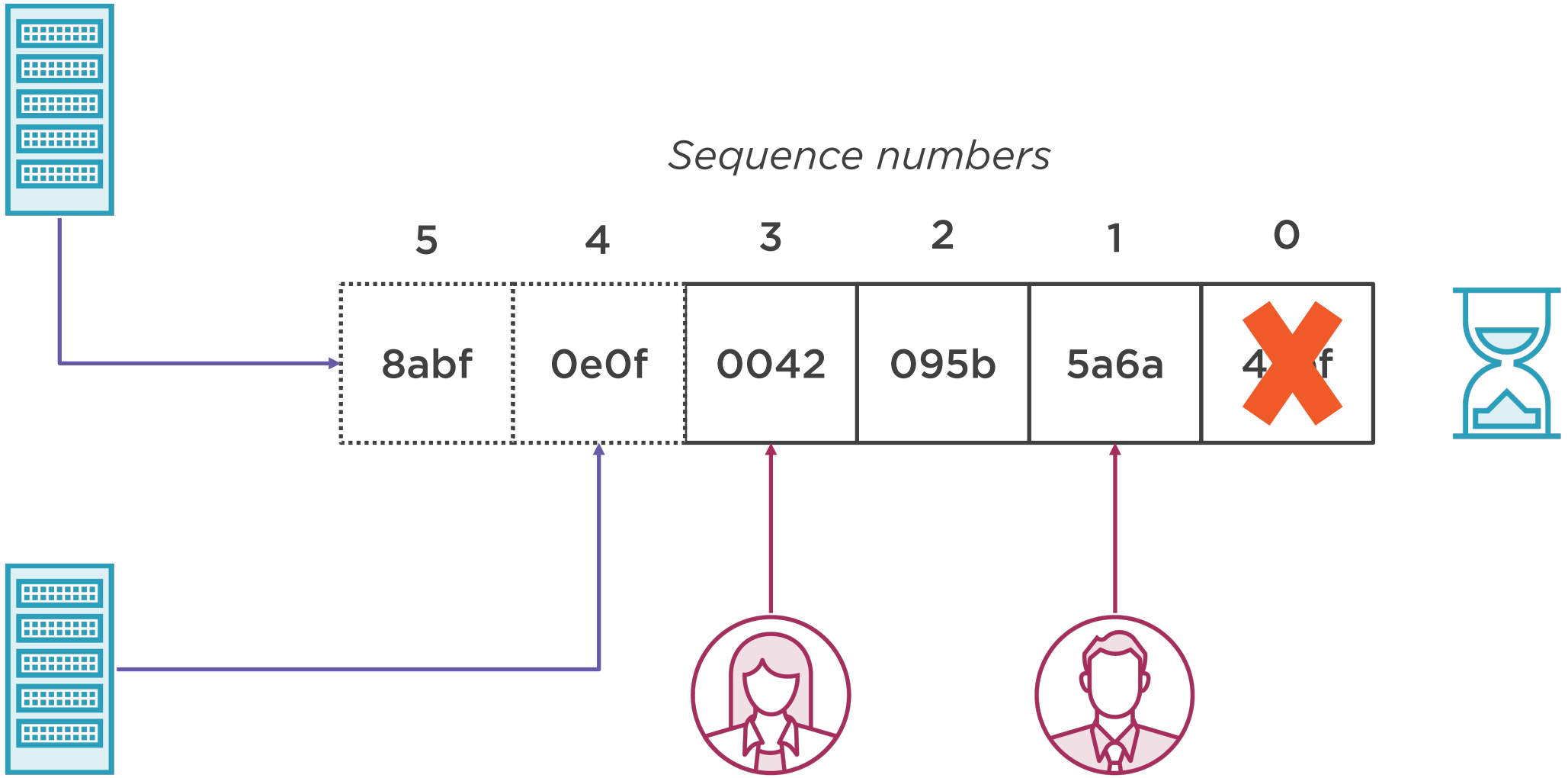
Reading and Writing Data to Kinesis

Ivan Mushketyk

@mushketyk brewing.codes

Kinesis Stream



Module Overview

Learn Kinesis API

Implement Kinesis producer

Implement Kinesis consumer

- Two different APIs to do this

Scale stream up and down

What Are We Going to Implement

Kinesis API

Stream

CreateStream

DeleteStream

ListStreams

DescribeStream

Manage Shards

MergeShards

SplitShard

Manage Stream

IncreateStreamRetentionPeriod

EnableEnhancedMonitoring

StartStreamEncryption

Write

PutRecord

PutRecords

Read

GetShardIterator

GetRecord

Fan-out Consumers

RegisterStreamConsumer

DeregisterStreamConsumer

ListStreamConsumers

```
{
   "StreamName": "tweetsStream"
   "PartitionKey": "NoSQL",
   "Data": "dba7..768=",
}
```

Writing Data to Kinesis

Writing a Tweet message to the "tweetsStream" stream with the "NoSQL" partition key

Reading Data from Twitter

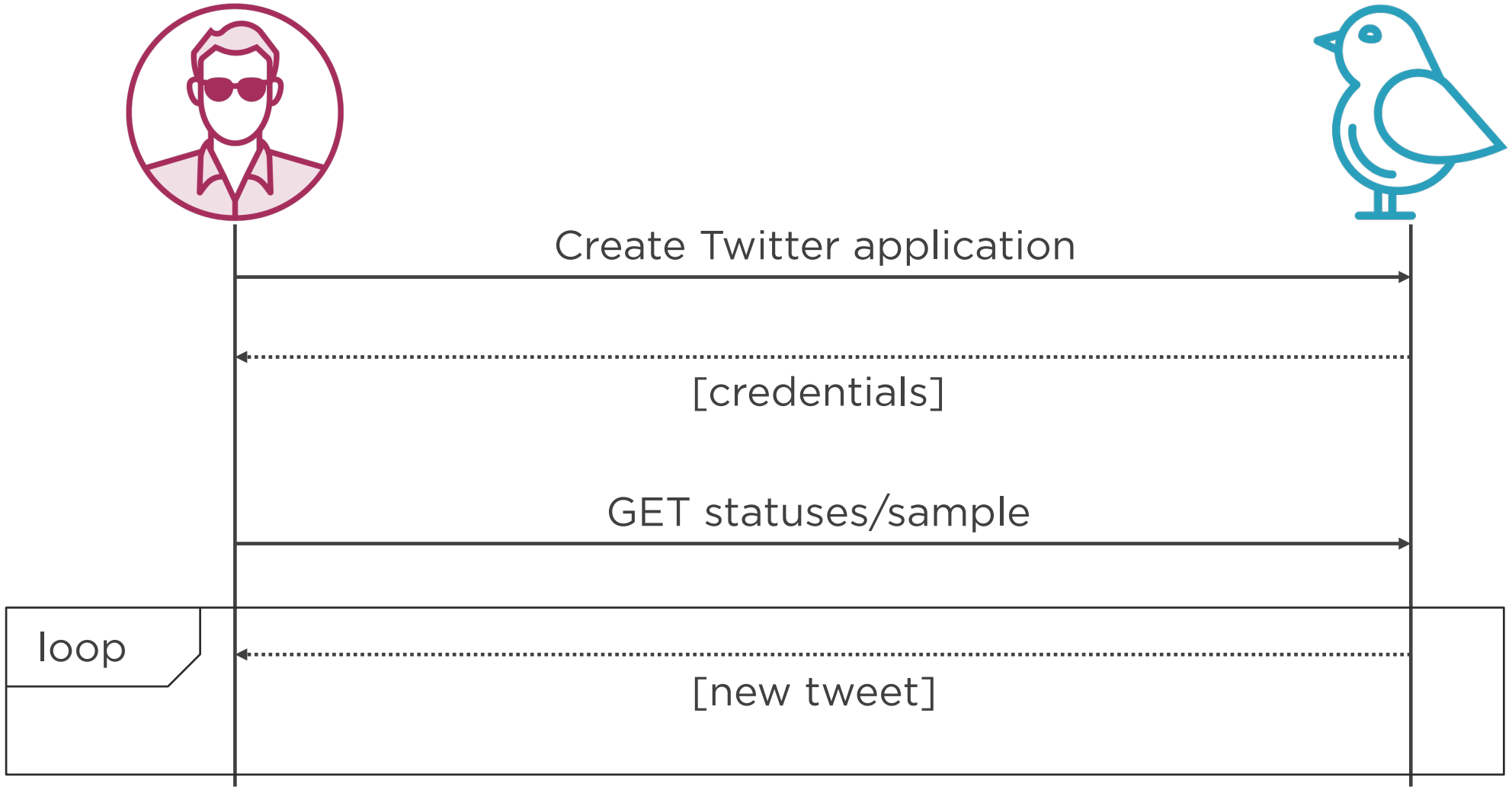
Read tweets from Twitter

- Create a Twitter application
- Use twitter4j

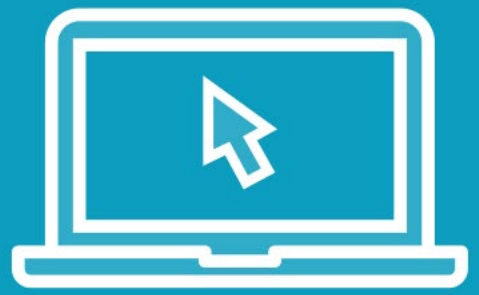
Main data source

Sample of tweets

Reading Data from Twitter



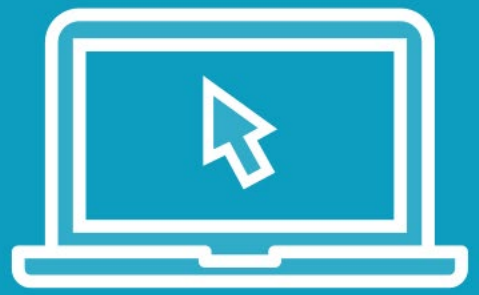
Demo



Create Twitter application

Read a stream of tweets

Demo

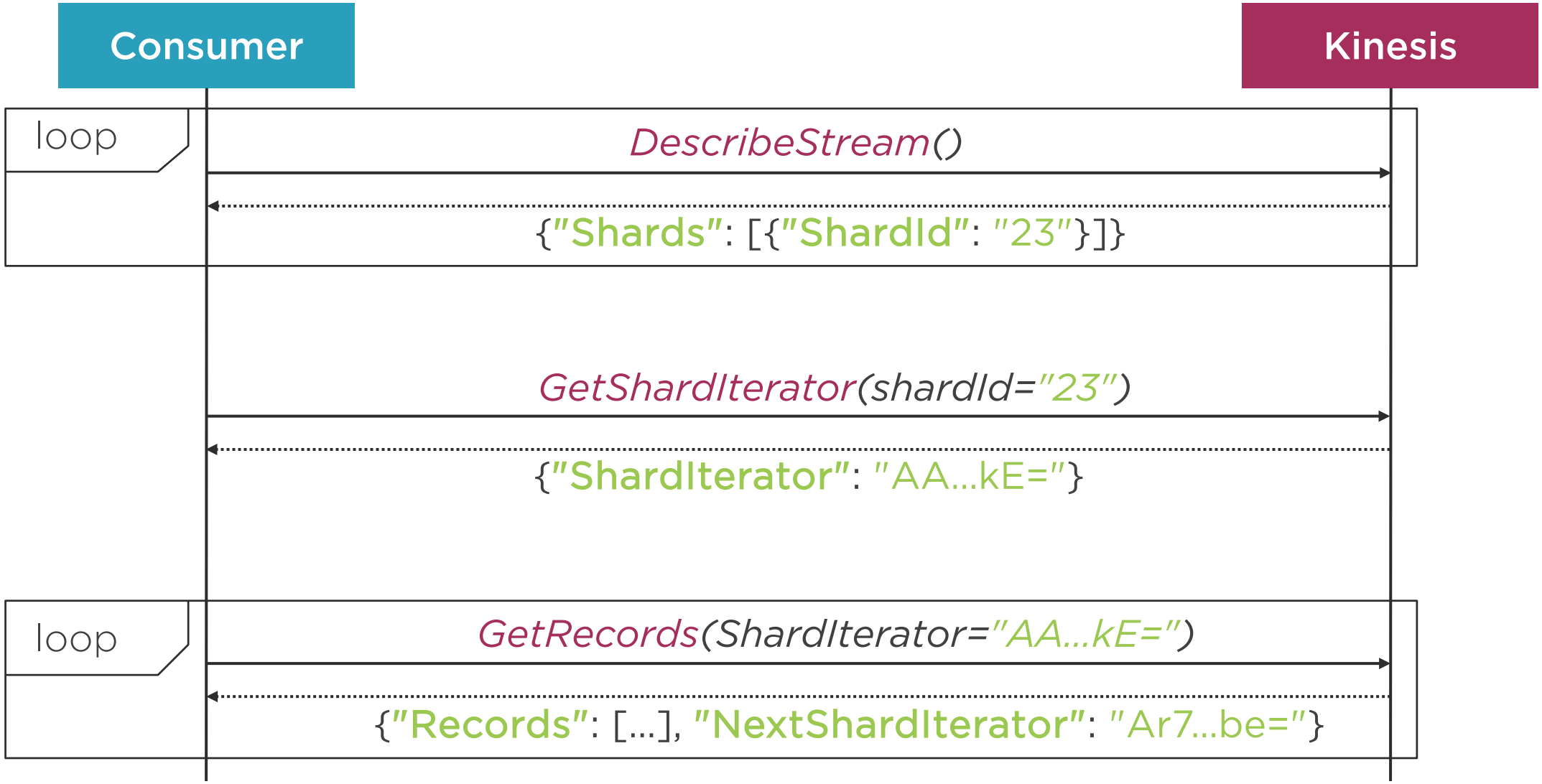


Implement Kinesis producer

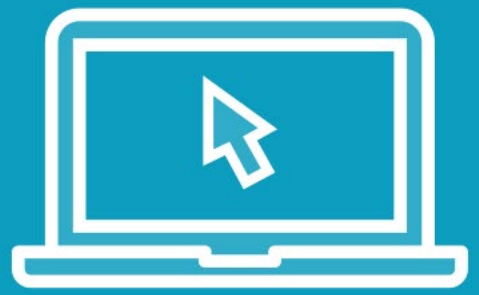
Write tweets data to Kinesis

Will read it from Kinesis later

Reading Data from Kinesis



Demo



Read tweets records from Kinesis

Write tweets to console

Consumer Implementation

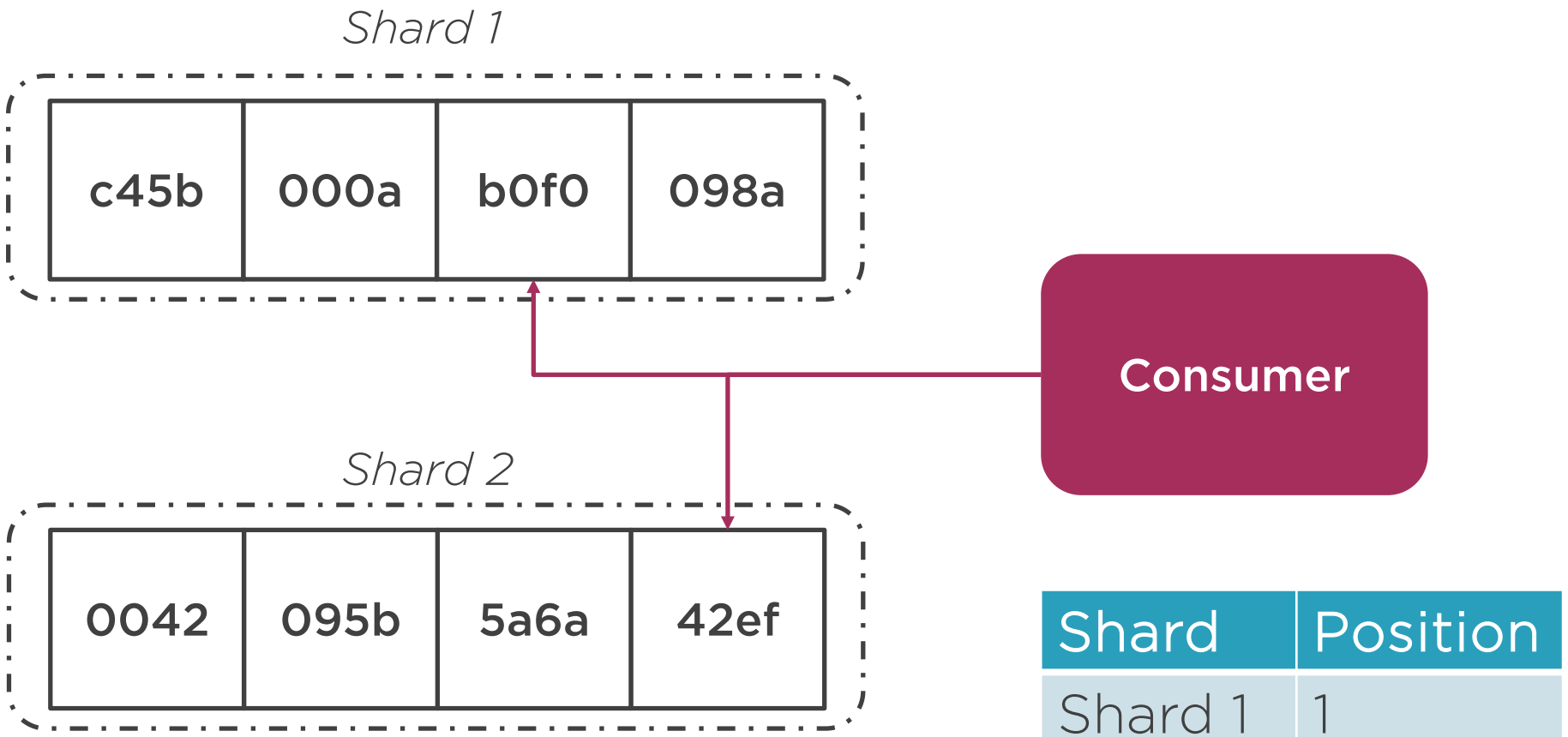
Correct consumer implementation

- Process all shards
- Assign shards to different workers
- Track progress in each shard
- Re-balance work if number of shard changes

No need to implement this ourselves

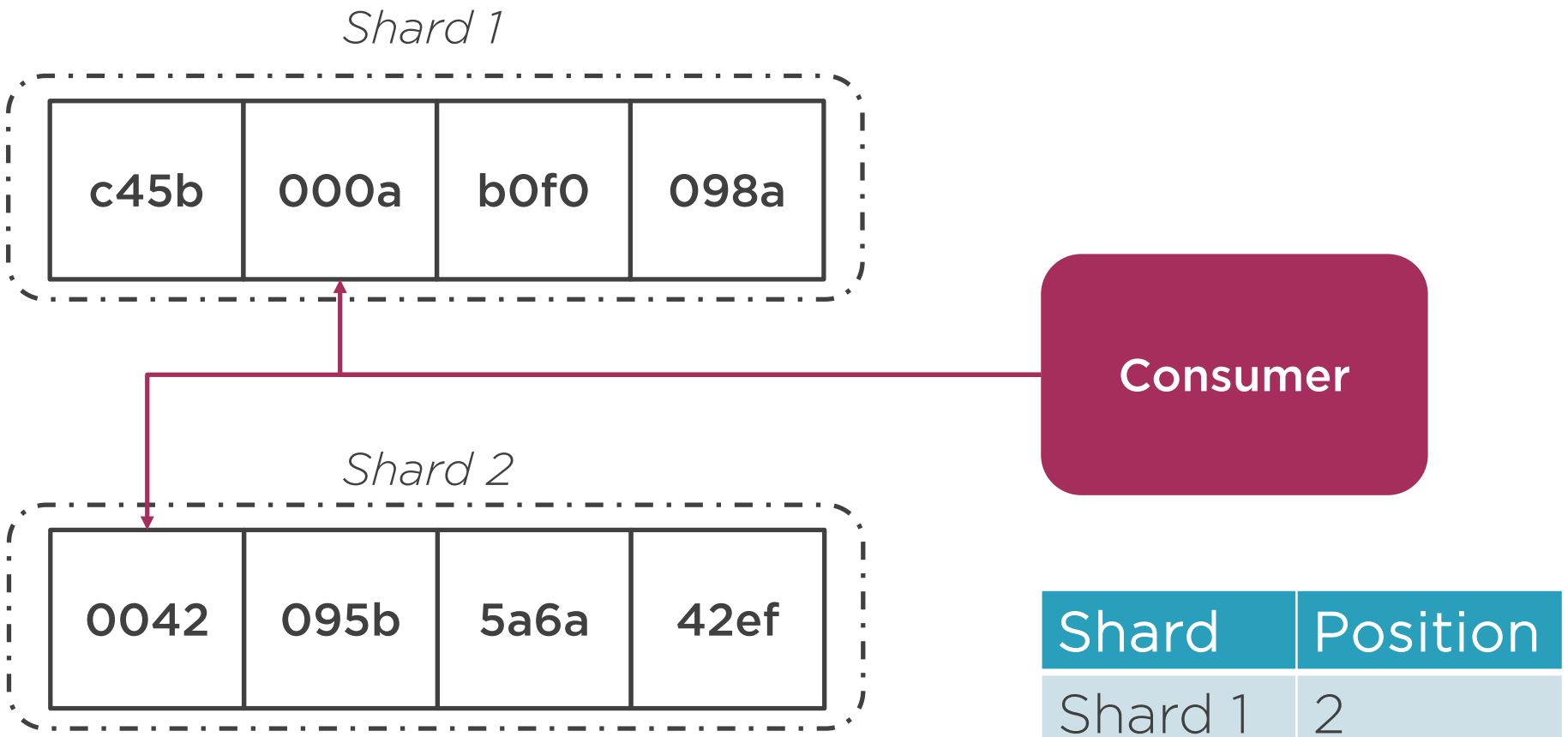
- Other libraries: e.g. KCL

Reading Data from Kinesis



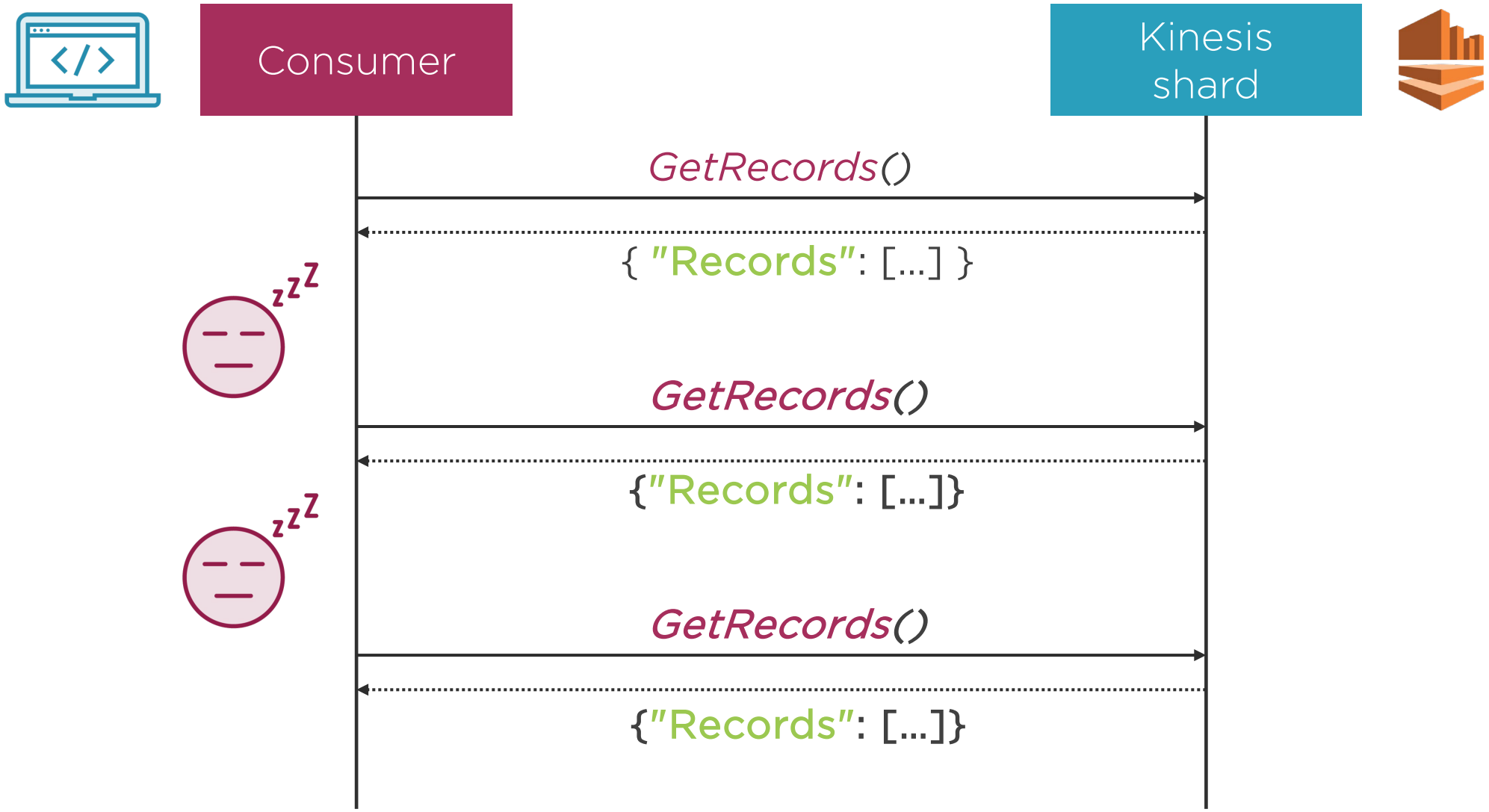
Shard	Position
Shard 1	1
Shard 2	0

Reading Data from Kinesis



Shard	Position
Shard 1	2
Shard 2	4

Reading from Kinesis



Shard Limitations

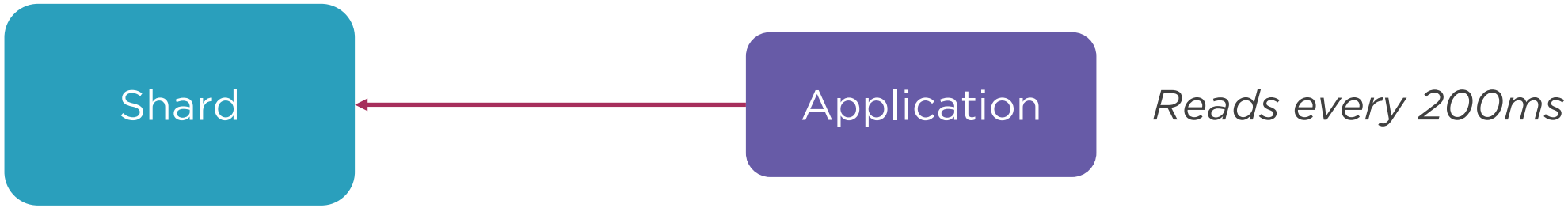
Amount of data

- One shard can return <2MB per second
- Divided among all consumers

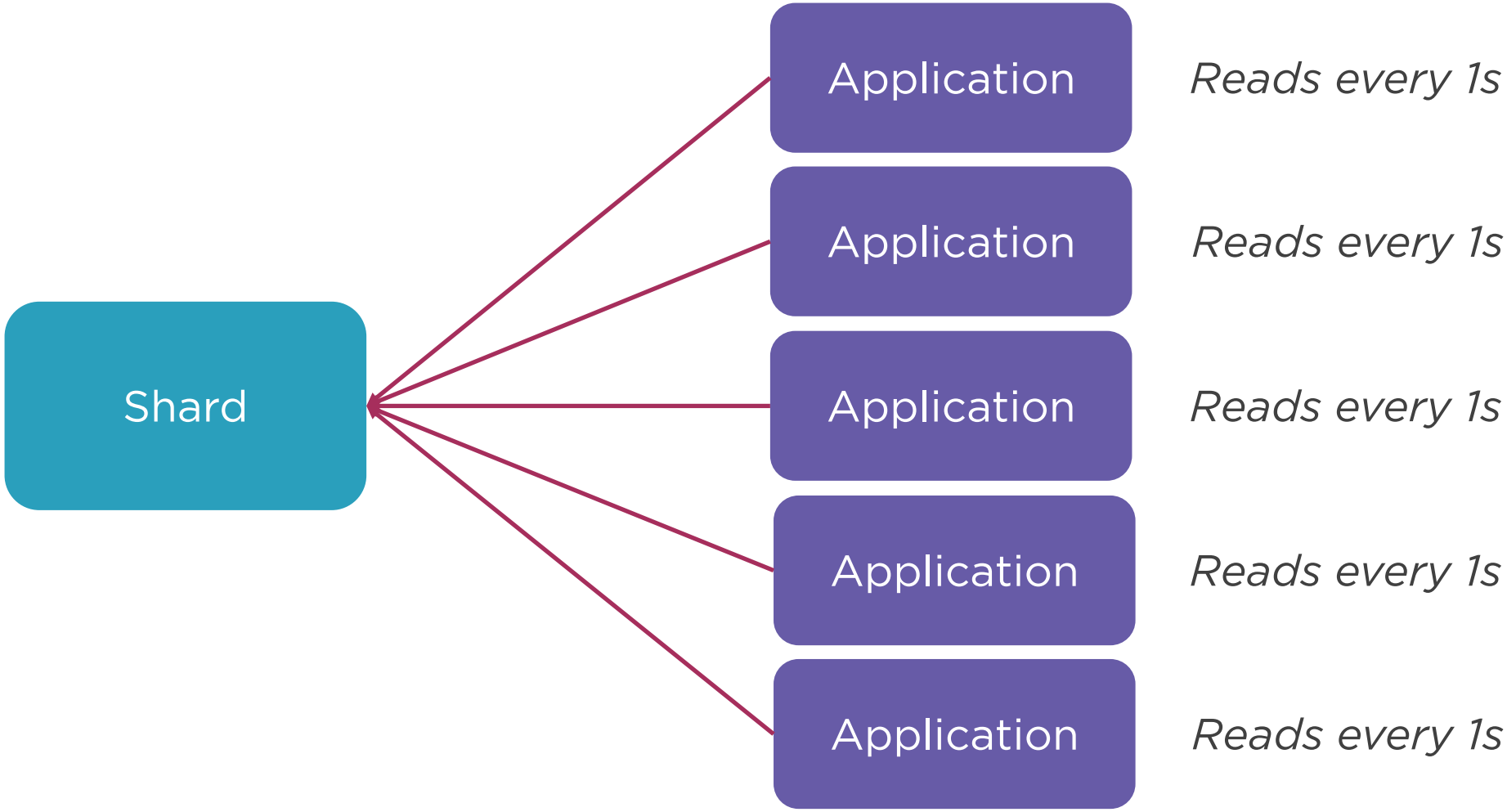
Number of read requests

- 5 reads per shard per second
- Can call `GetRecords()` at most every 200 ms

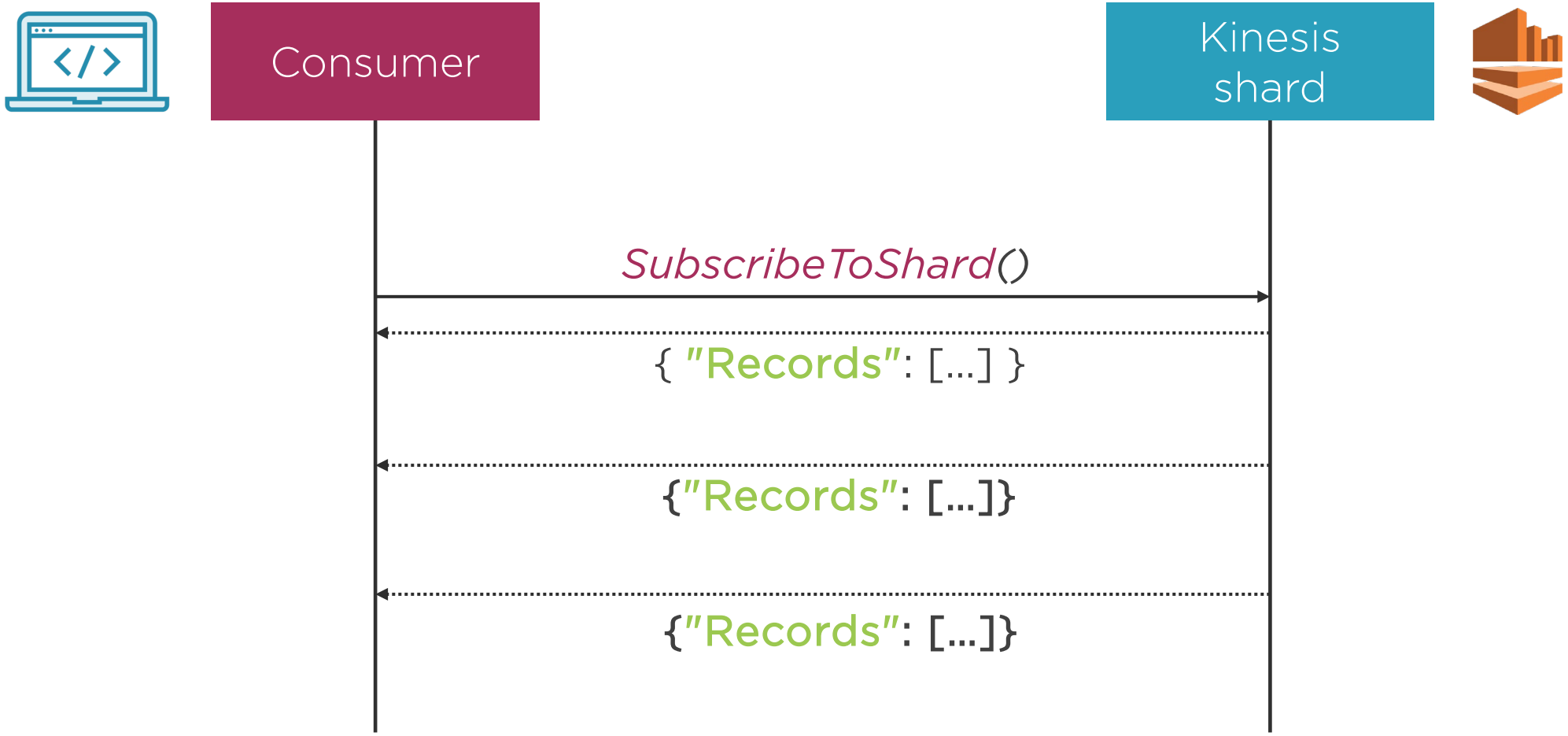
Number of Read Requests



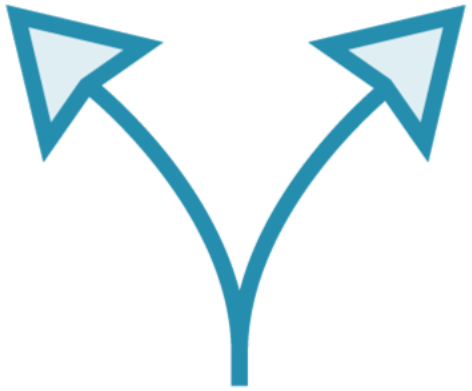
Number of Read Requests



Enhanced Fan-Out Consumer



Enhanced Fan-Out Consumer



Amount of data

- Up to 2MB per second per consumer

Lower latency

- Around 70 ms

Number of consumer

- Up to 5 (not a hard limit)

Extra cost

- Per GB retrieved and per shard

SDK for Java 2.x

Fan-out consumer only for AWS SDK 2.x

- We were using SDK 1.x in this course
- Used in most existing apps

AWS SDK 2.x

- Major SDK rewrite
- Built for Java 8+

```
CompletableFuture<Record> future = getRecord();
// Block until a result is ready
future.get();
// Do not block. Wait for result to be ready
future.thenApply(r -> processRecord(r));
```

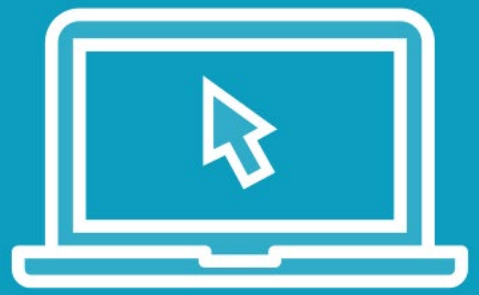
Java 8 CompletableFuture

Class that represents a future result (e.g. HTTP request)

Can process results/errors asynchronously.

Can wait until a result is ready

Demo



Read tweets stream

Implement enhanced fan-out consumer

Using Kinesis Efficiently

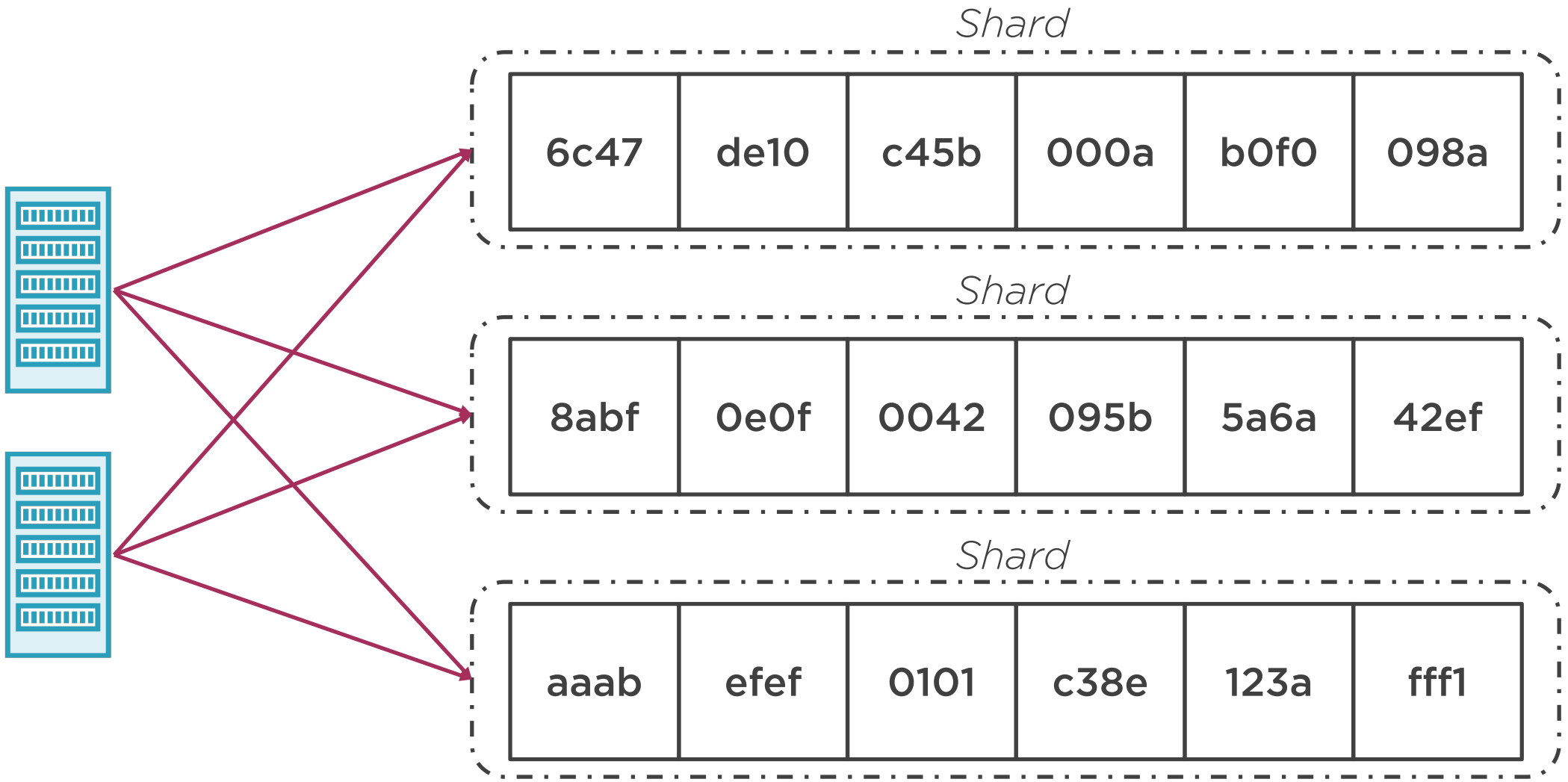
A shard can process limited number of requests

- Will return throughput exceed exception
- **Solution:** use more shards

You can have more shards than necessary

- Will pay more than necessary
- **Solution:** reduce number of shards

Scaling Kinesis Stream



Scaling Kinesis API

SplitShard

Replace a single shard with two shards

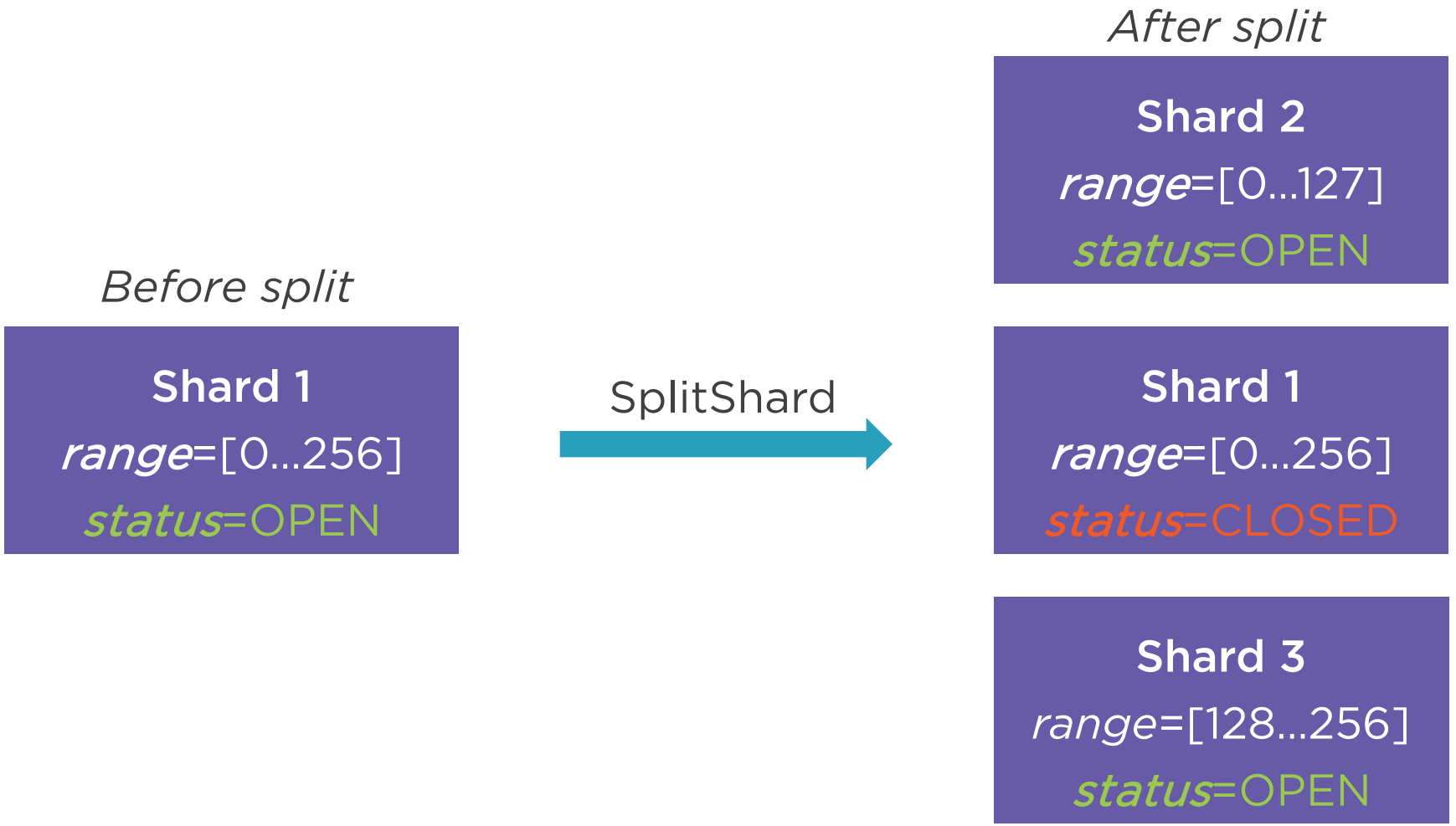
MergeShards

Replace two shards with a single shard

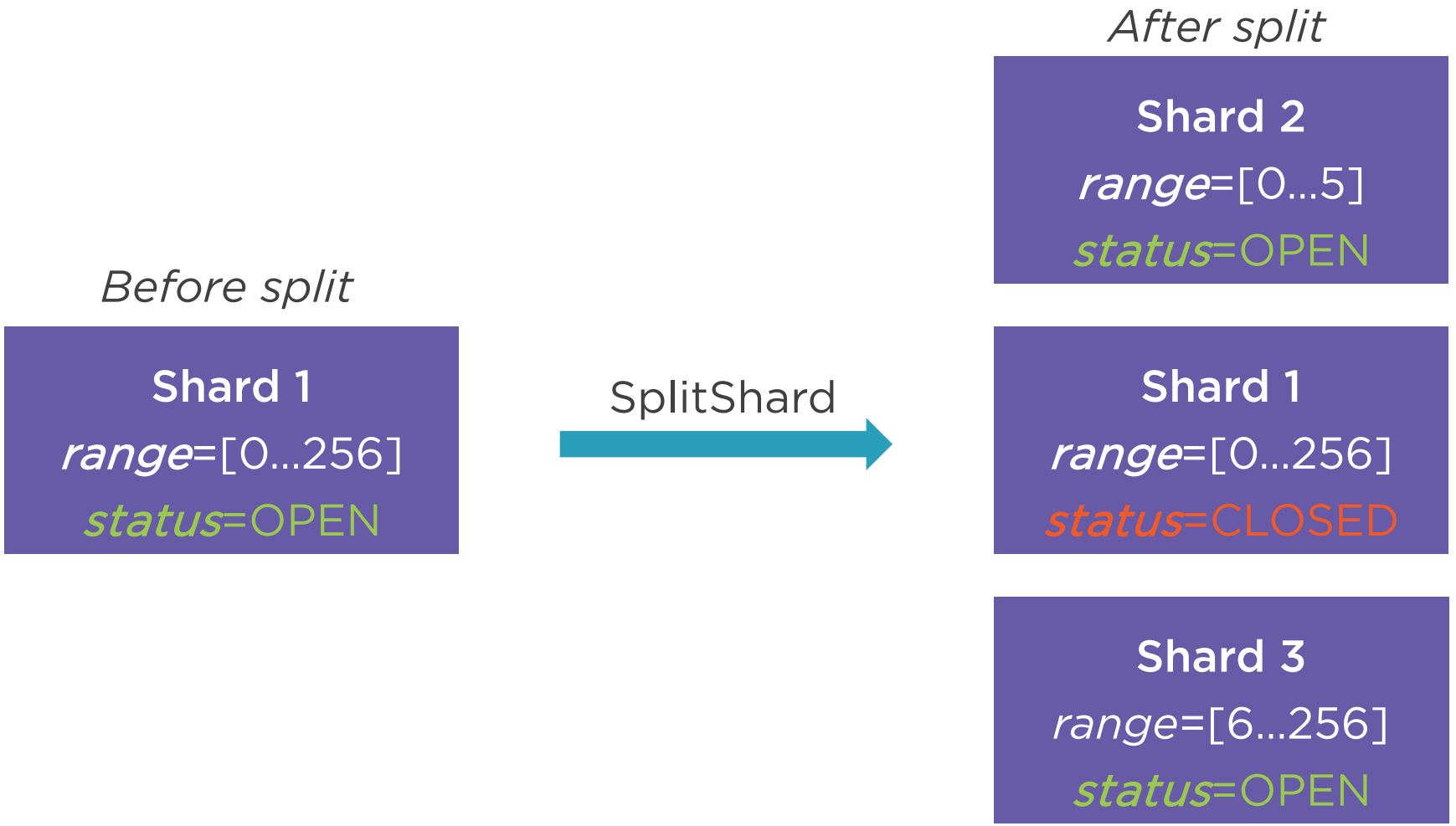
UpdateShardCount

Scale up or down to the specified number of shards

Splitting a Shard



Splitting a Shard Unevenly

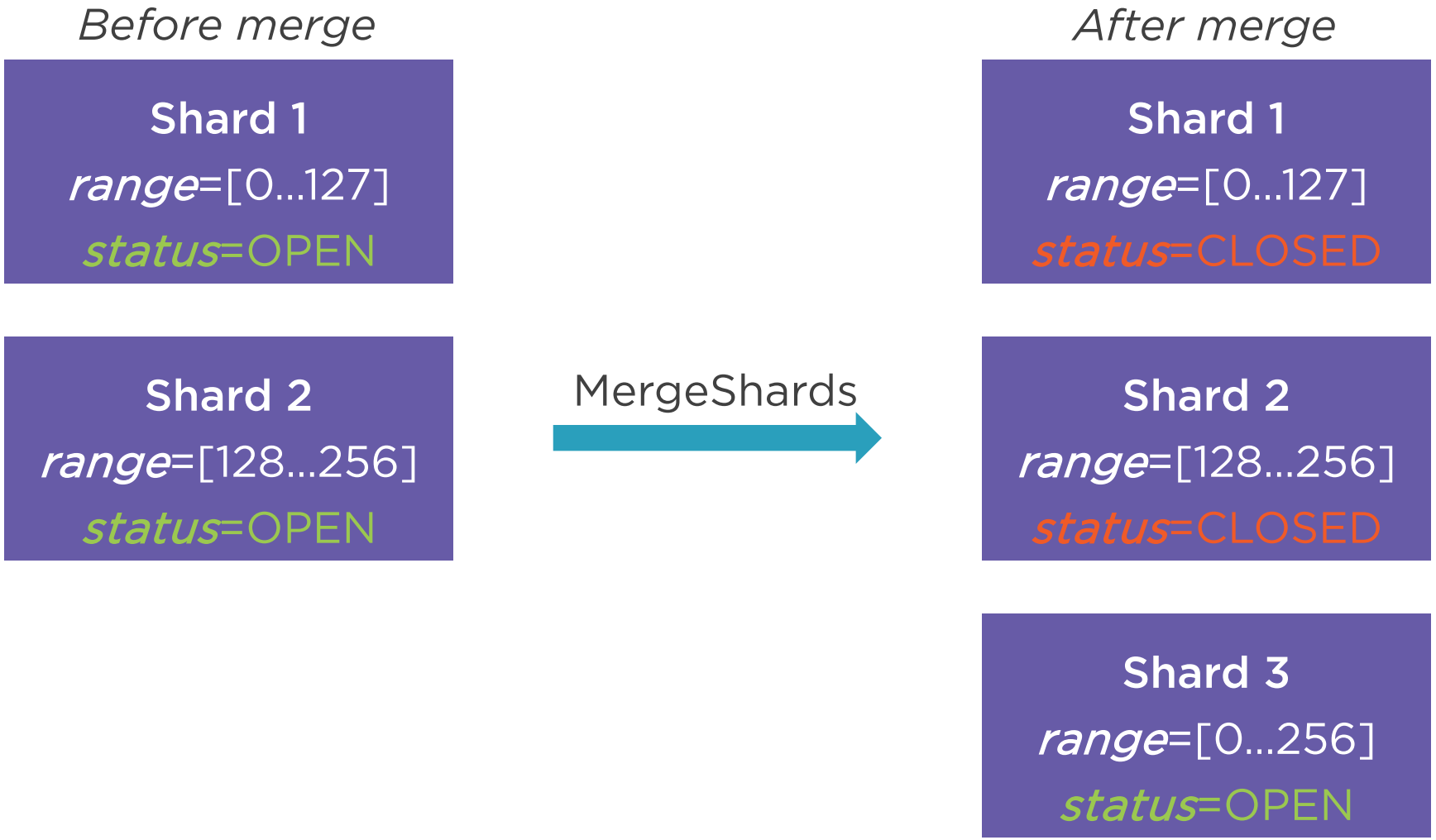


```
{
  "NewStartingHashKey": "127",
  "ShardToSplit": "shard-23",
  "StreamName": "tweetsStream"
}
```

SplitShard API Call

Splits shard "shard-23" into two shards.

Merging Shards



```
{
  "AdjacentShardToMerge": "shard-2",
  "ShardToMerge": "shard-3",
  "StreamName": "tweetsStream"
}
```

MergeShards API Call

Merges shard "shard-3" into "shard-2" and join their ranges

Returns HTTP 200

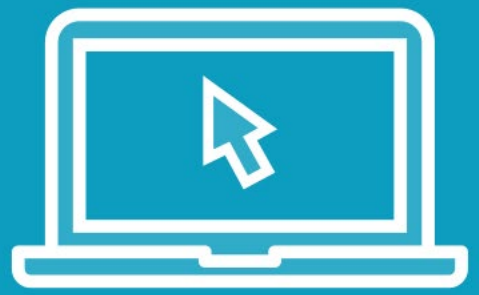
Shards must have adjacent ranges

```
{
  "ScalingType": "UNIFORM_SCALING",
  "StreamName": "tweetsStream",
  "TargetShardCount": 100
}
```

Setting Number of Shards

Change number of shards to 100.

Demo



Scale-up Kinesis stream

Scale-down Kinesis stream

Kinesis Limits

Data storage

Write per shard

Read from a shard

Data Storage Limitations

Number of shards:

- Up to 500 shards in Virginia, Oregon, Ireland
- Up to 200 shards elsewhere
- No a hard limit

Data retention period:

- 24 hours by default
- Can be up to 168 hours
- Hard limit

Data Writing Limitations

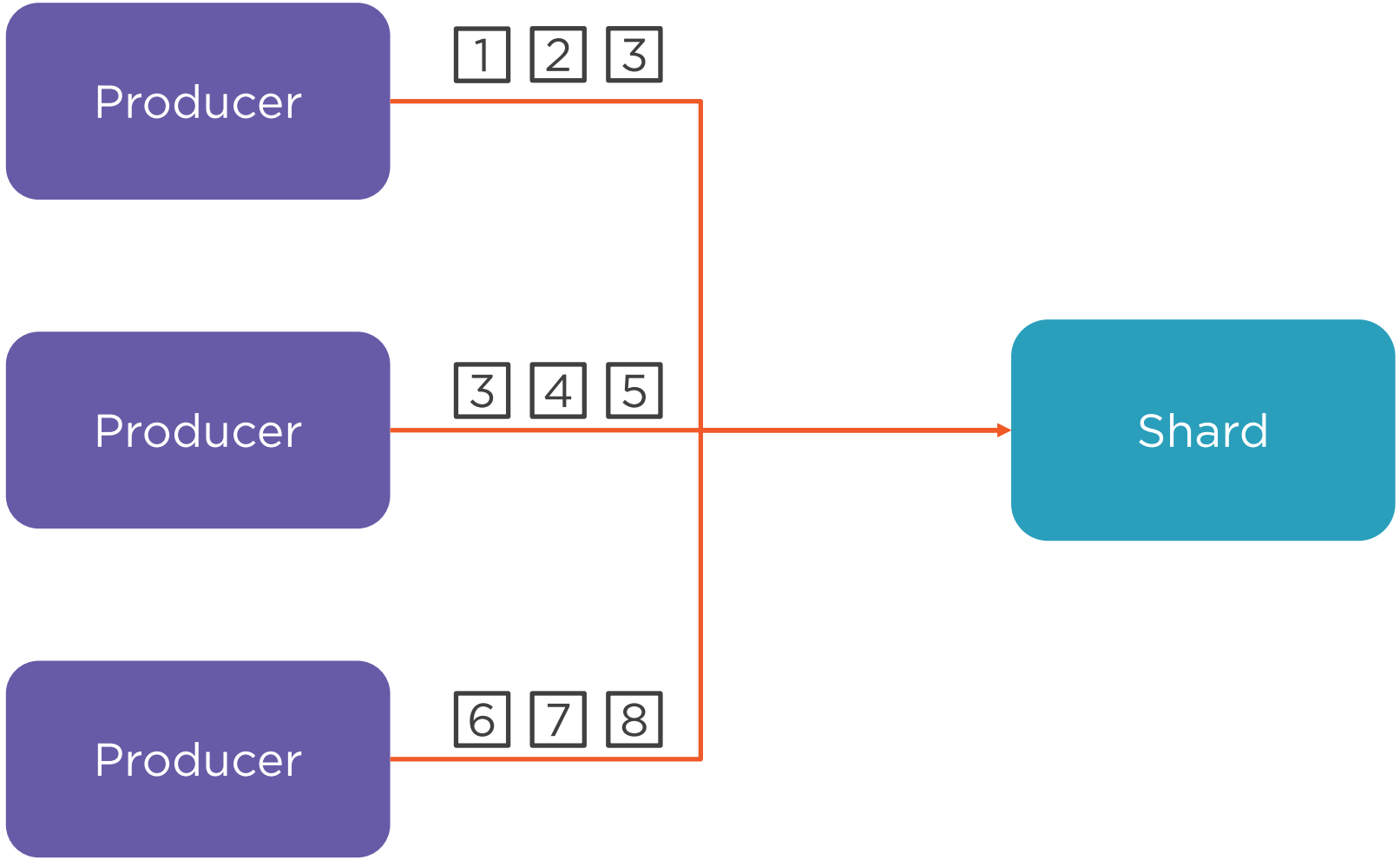
Shard throughput

- Up to 1MB
- **Solution:** Scale up

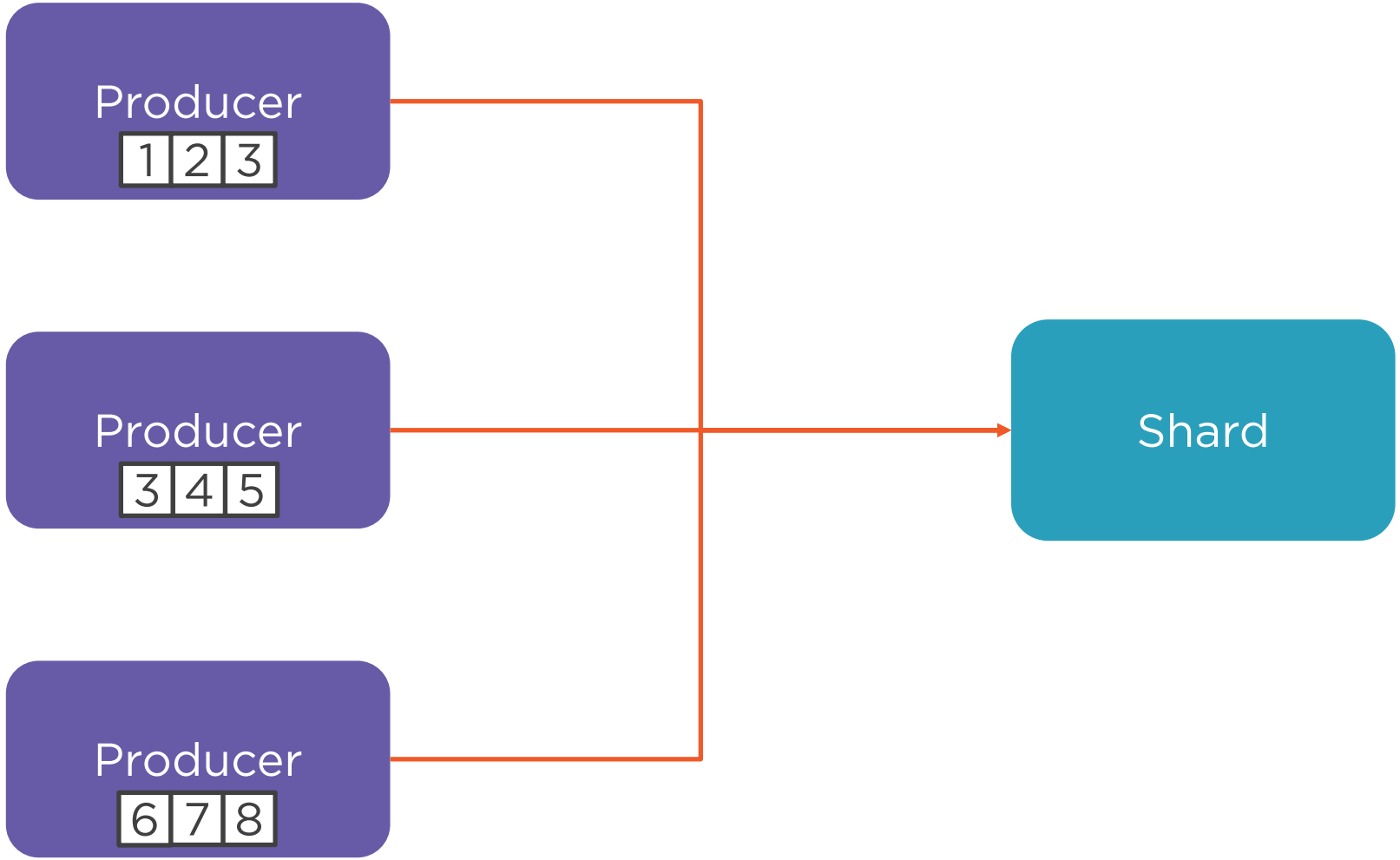
Number of transactions

- Up to 1,000 records per second
- **Solution:** Producer buffering

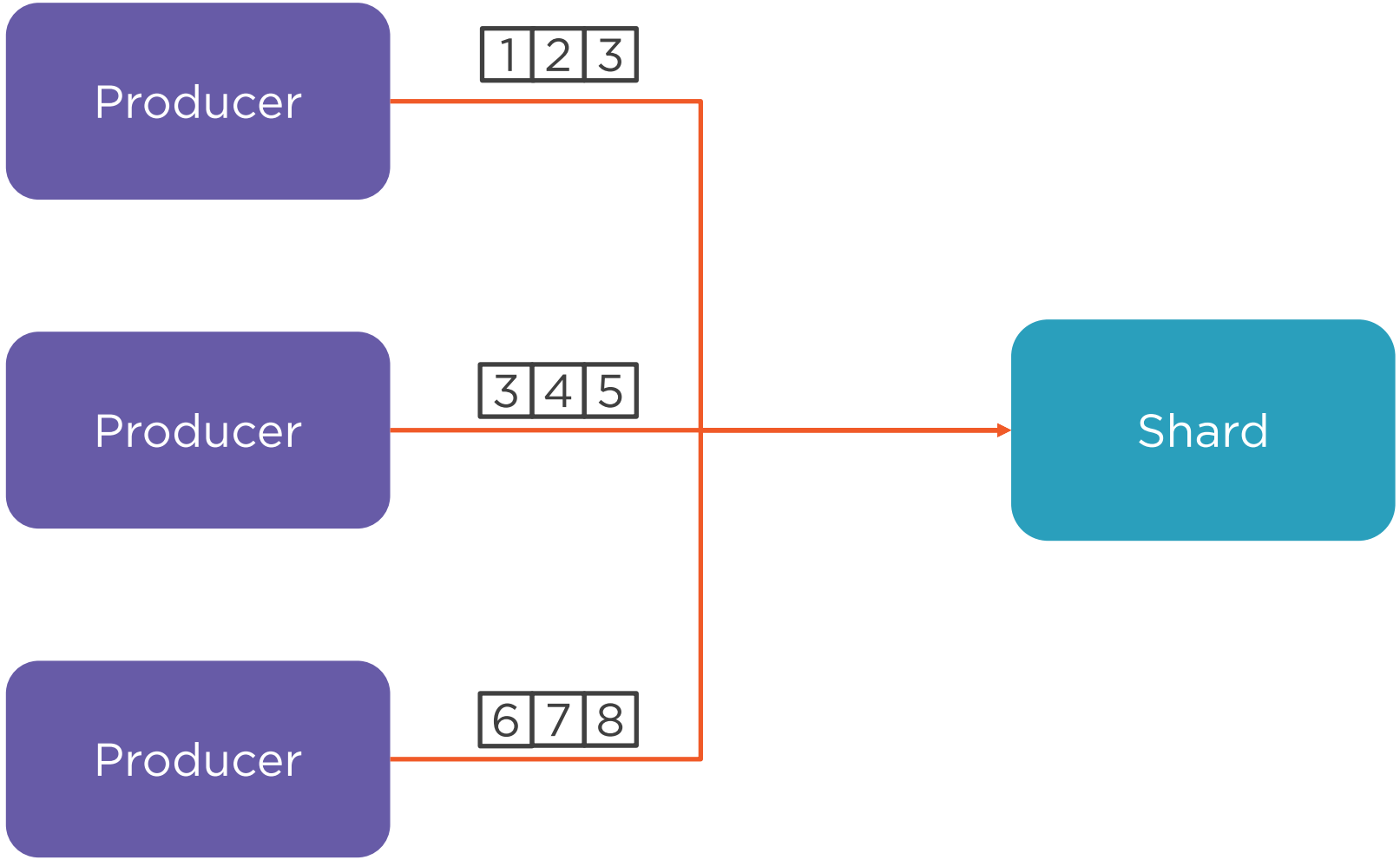
Write Request per Record



Write Request per Record



Write Request per Record



Read Limitations

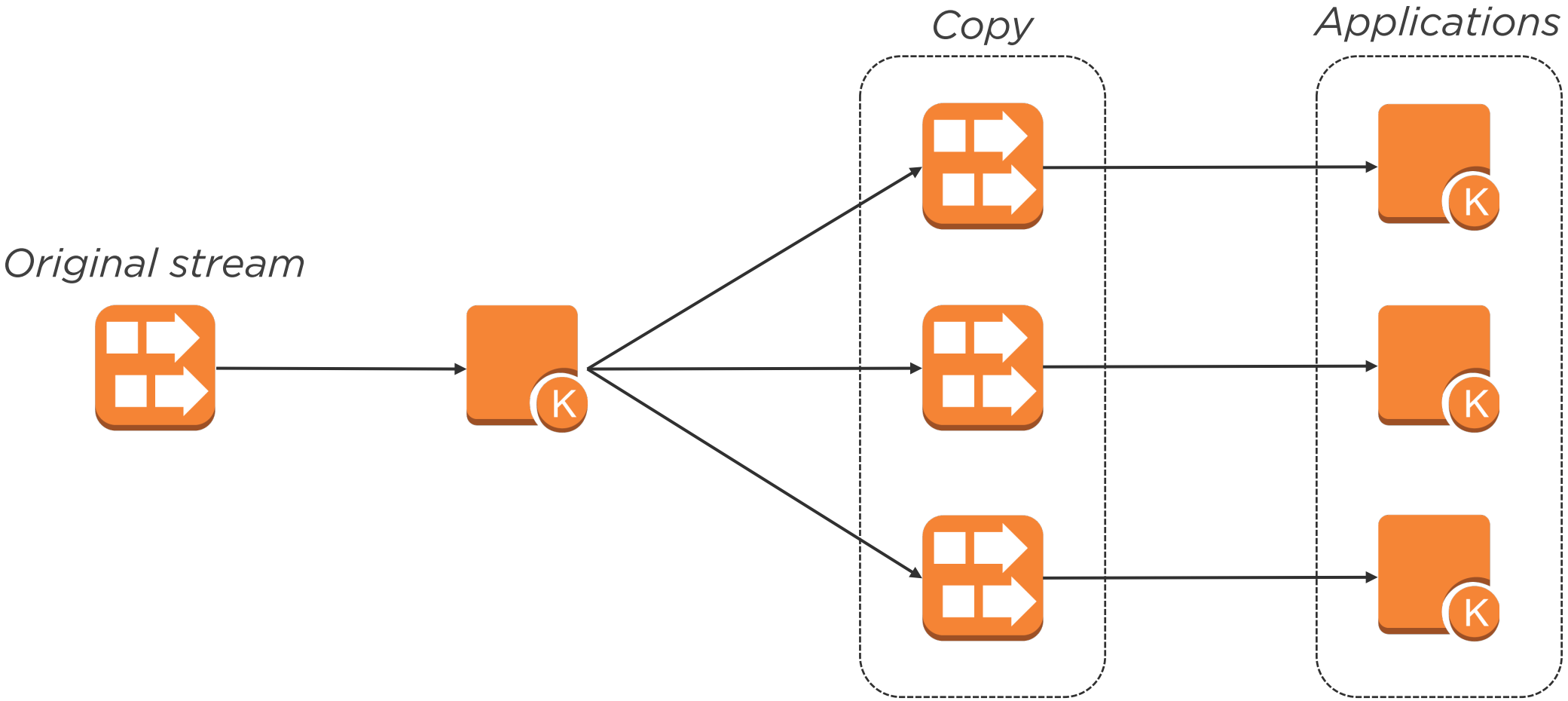
Amount of data

- GetRecords returns <10MB
- One shard can return <2MB per second
- **Solution:** Scale up

Number of transactions

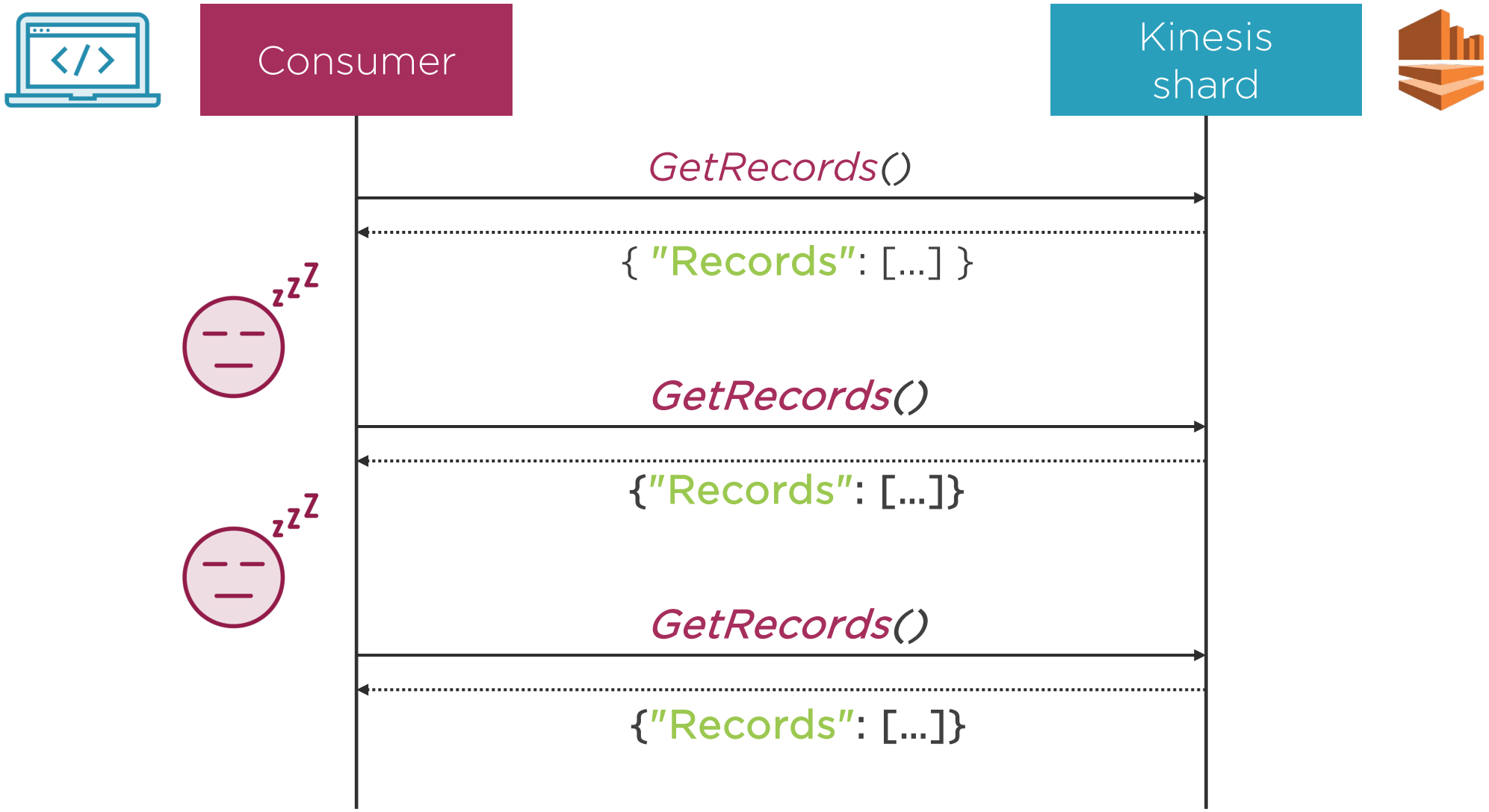
- 5 reads per shard per second
- **Solution:** Copy Kinesis data

Increasing Number of Transactions

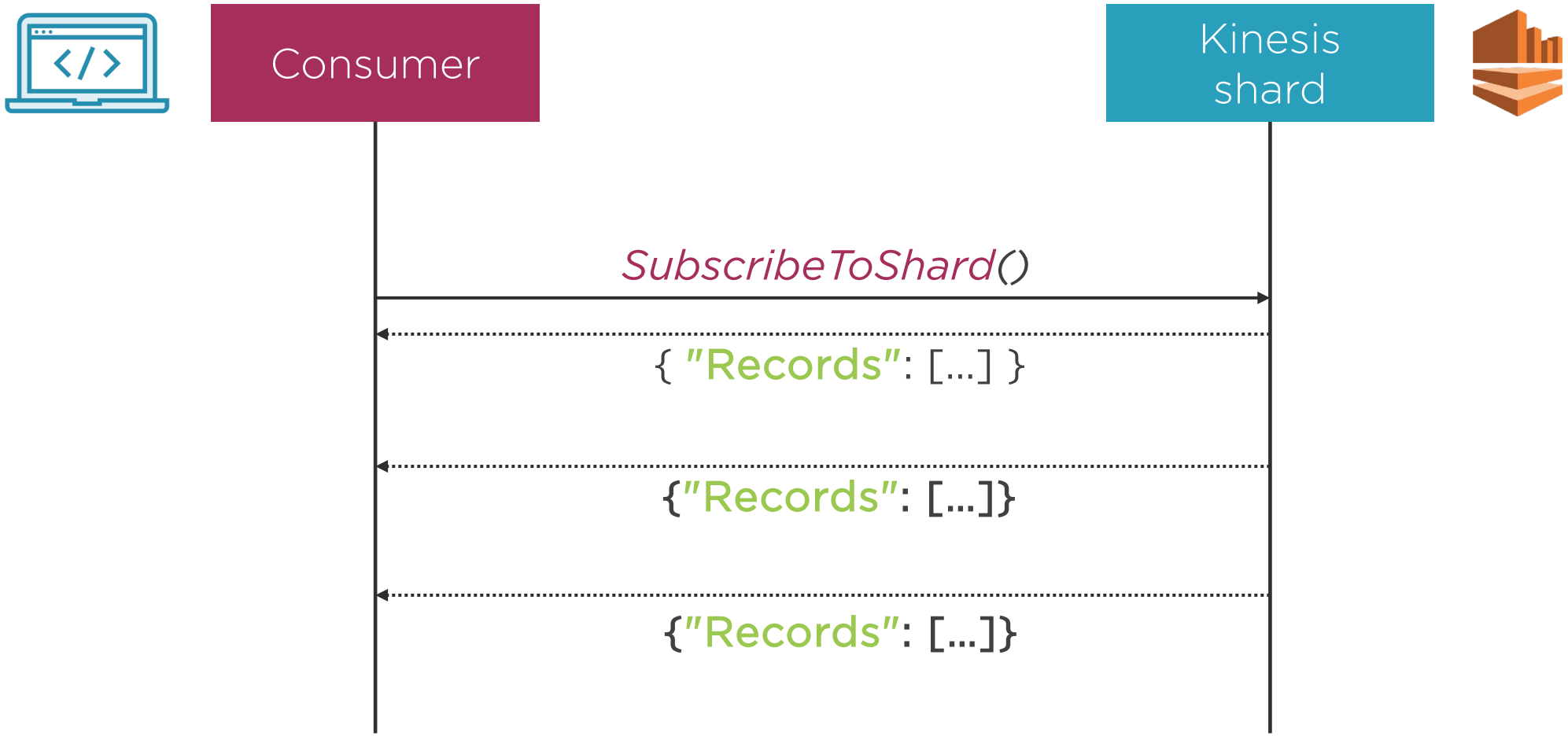


Module Summary

Module Summary



Module Summary



Module Summary Cont.