

Stream processing with R and Amazon Kinesis

EARL 2016 London



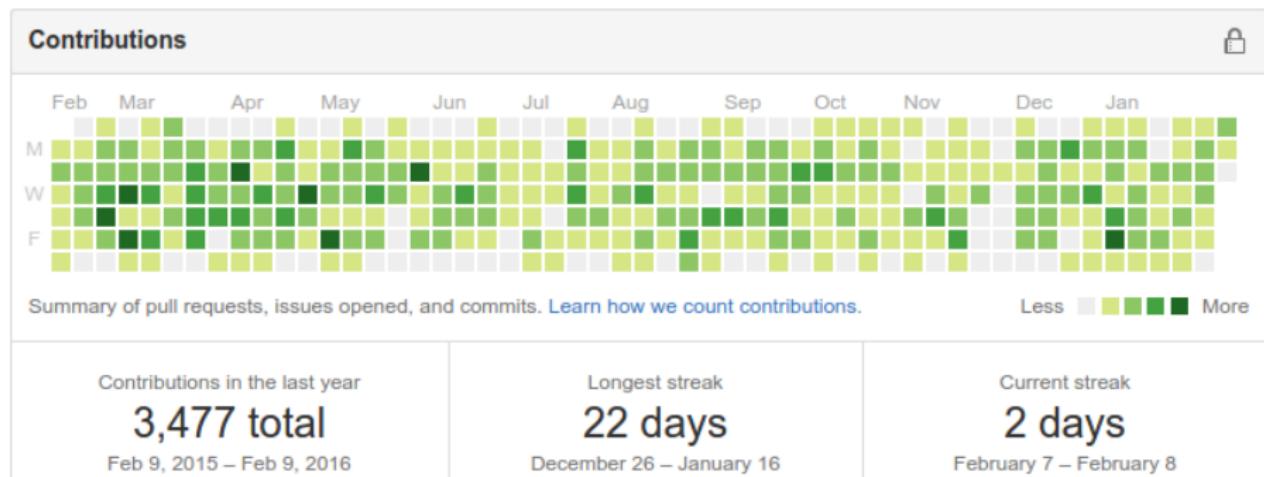
CARD.COM

Gergely Daroczi

@daroczig

September 15 2016







Gergely Daróczi @daroczig · Apr 11

Just received my "I ❤️ R" prepaid debit card from @CARD. Will be fun to use this #rstats designed card at #user2015 :)

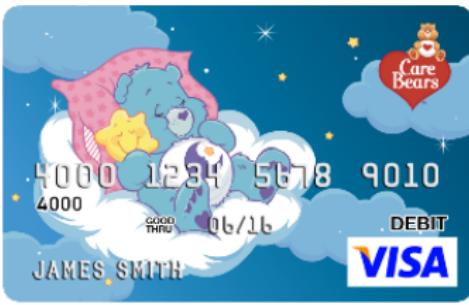


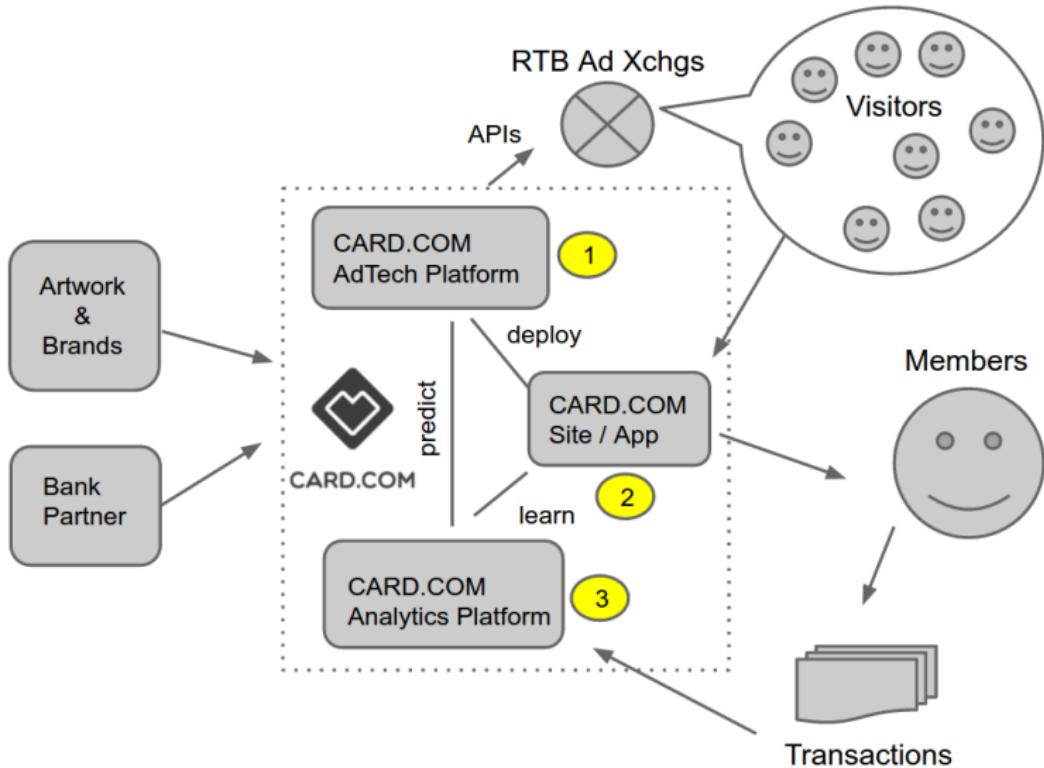
RETWEETS FAVORITES

10

16







- card transaction processors
- card manufacturers
- CIP/KYC service providers
- online ad platforms
- remarketing networks
- licensing partners
- communication engines
- others



Why R?

User Defined Java Class

Step name: ShortenDate

Classes and code fragments:

- Classe
- Code Snippets
- Input fields
- Getting fields...please wait
- Info fields
- Getting fields...please wait
- Output fields
- Getting fields...please wait

Class code

```
Processor
import java.text.SimpleDateFormat;
import java.util.Date;
import java.text.ParseException;
import java.util.TimeZone;

private SimpleDateFormat df1 = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss,SSS");

private SimpleDateFormat df2 = new SimpleDateFormat("yyyy-MM-dd HH");

public boolean processRow(StepMetaInterface smi, StepDataInterface sdi) throws KettleException, ParseException
{
    Object[] r = getRow();
    if (r == null) {
        setOutputDone();
        return false;
    }

    if (first)
    {
        first = false;
    }

    // It is always safest to call createOutputRow() to ensure that your output row's Object[] is large
    // enough to handle any new fields you are creating in this step.
    r = createOutputRow(r, data.outputRowMeta.size());

    df2.setTimeZone(TimeZone.getTimeZone("America/Los_Angeles"));

}
```

Line #: 0

Fields | Parameters | Info steps | Target steps

Fields

#	Fieldname	Type	Length	Precision
1	RPT_DATE_SHORT	String		

Clear the result fields?

Help OK Cancel Test class

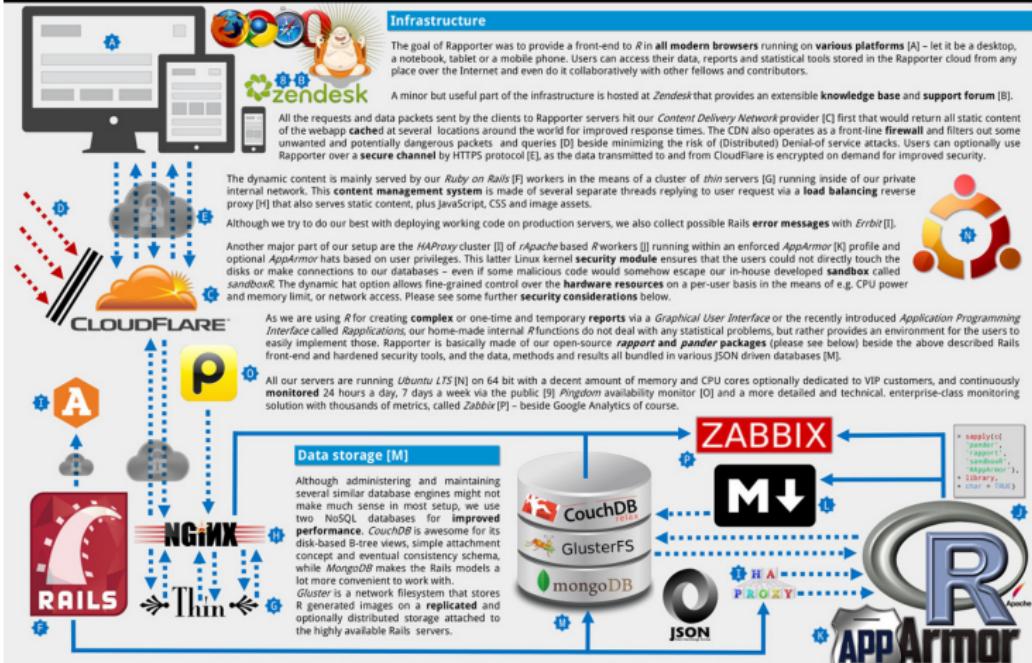


P22

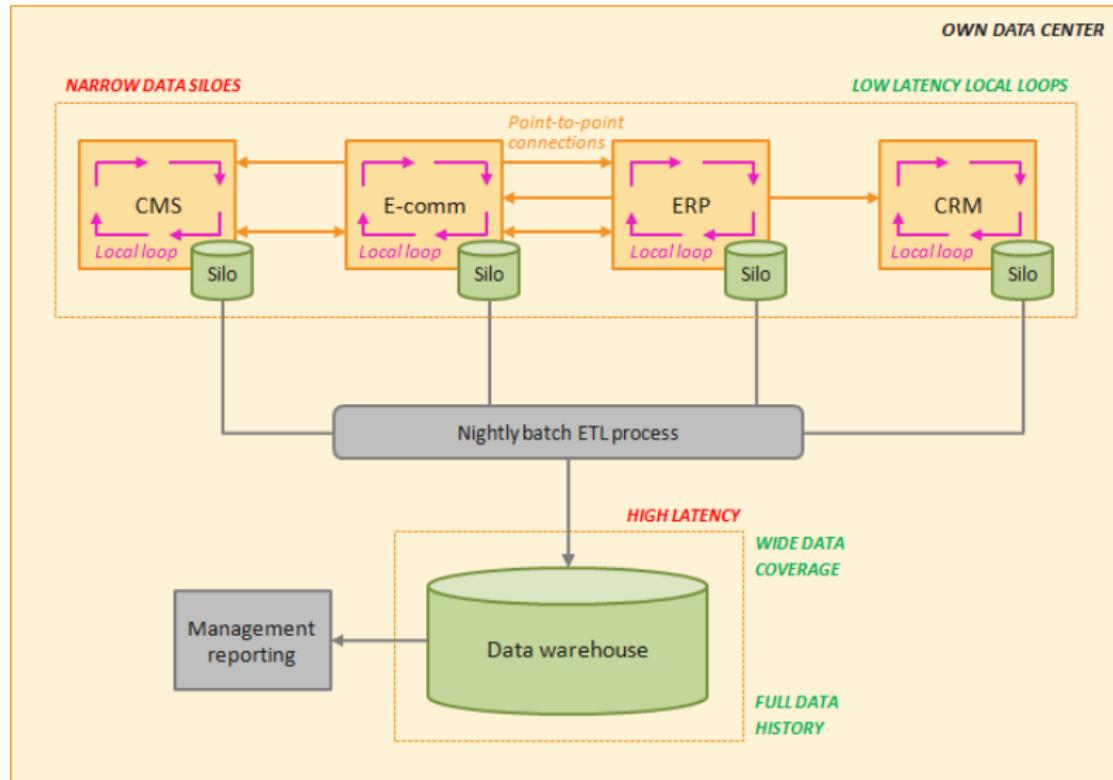
A cloud infrastructure for R reports

Gergely Darócz1,2,3; Aleksandar Blagotic1,2

Founder, Easystats Ltd, UK | Assistant lecturer, PPKE BTK, HUN | PhD candidate, Corvinus University, HUN | R/web-developer, Easystats Ltd, UK | Msc student, University of Niš, SRB

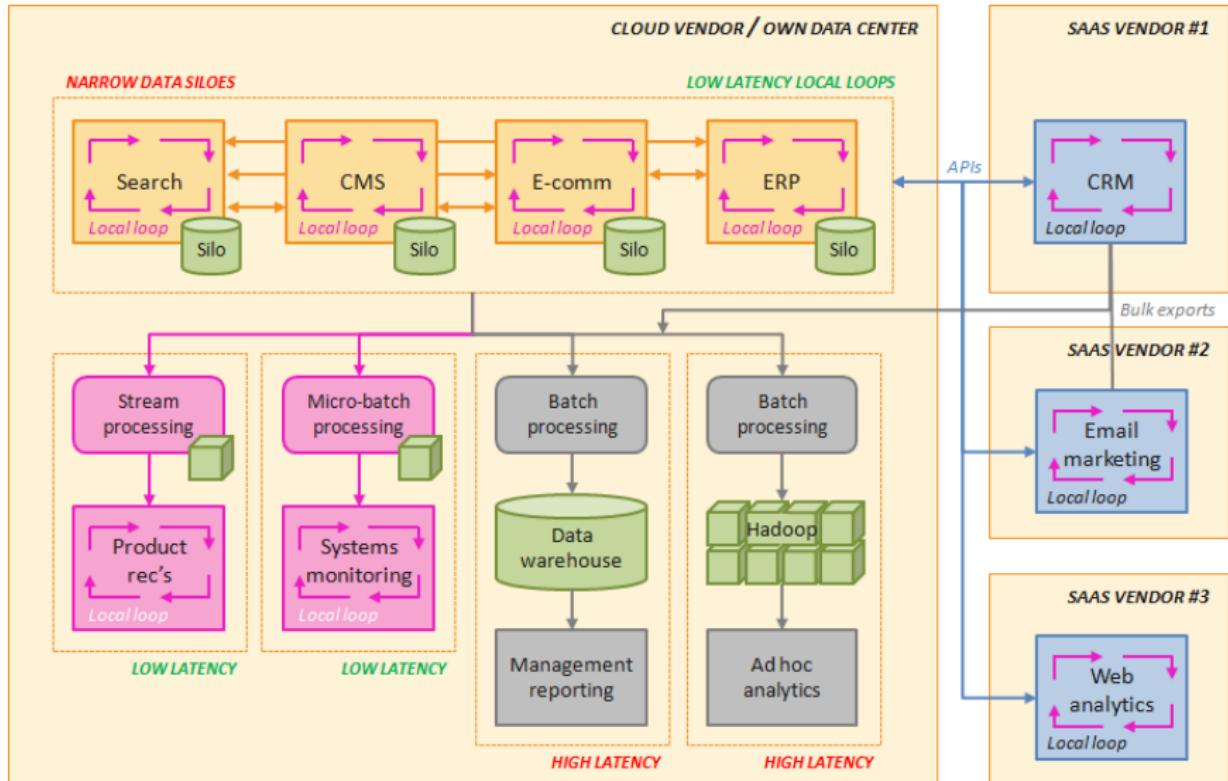






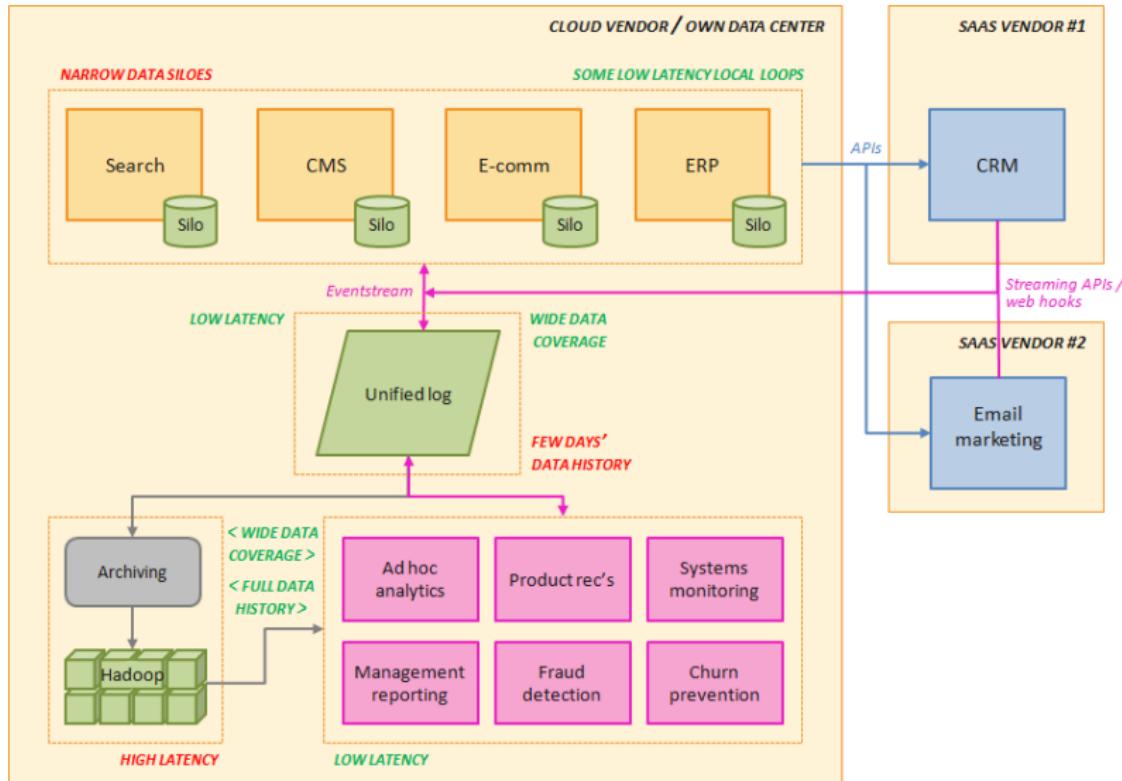
Source: SnowPlow

The Hybrid Era



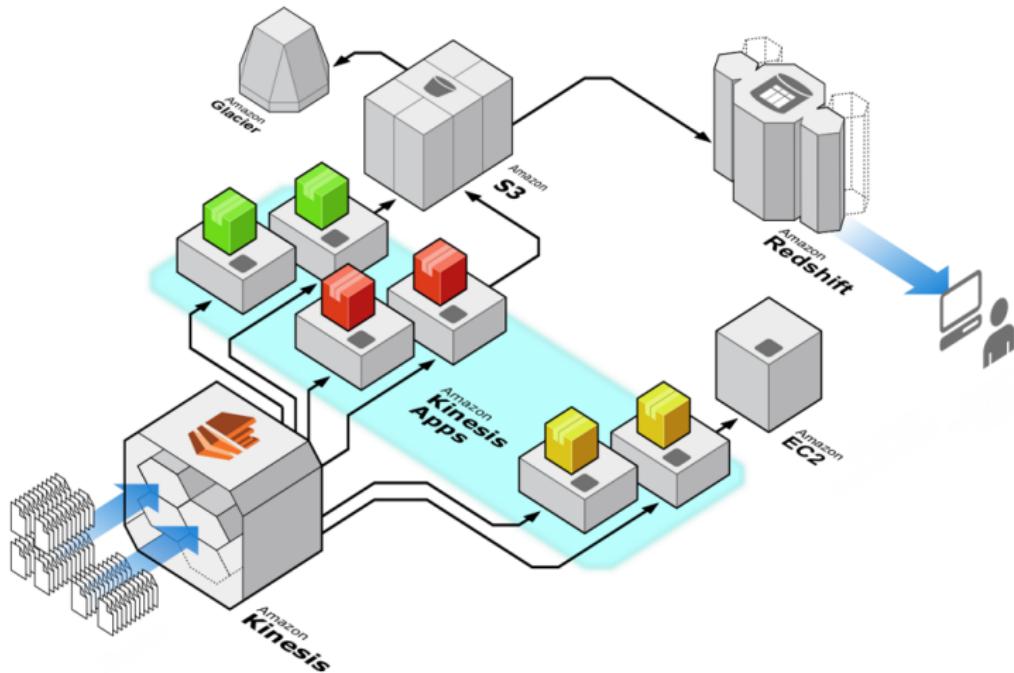
Source: SnowPlow

The Unified Era



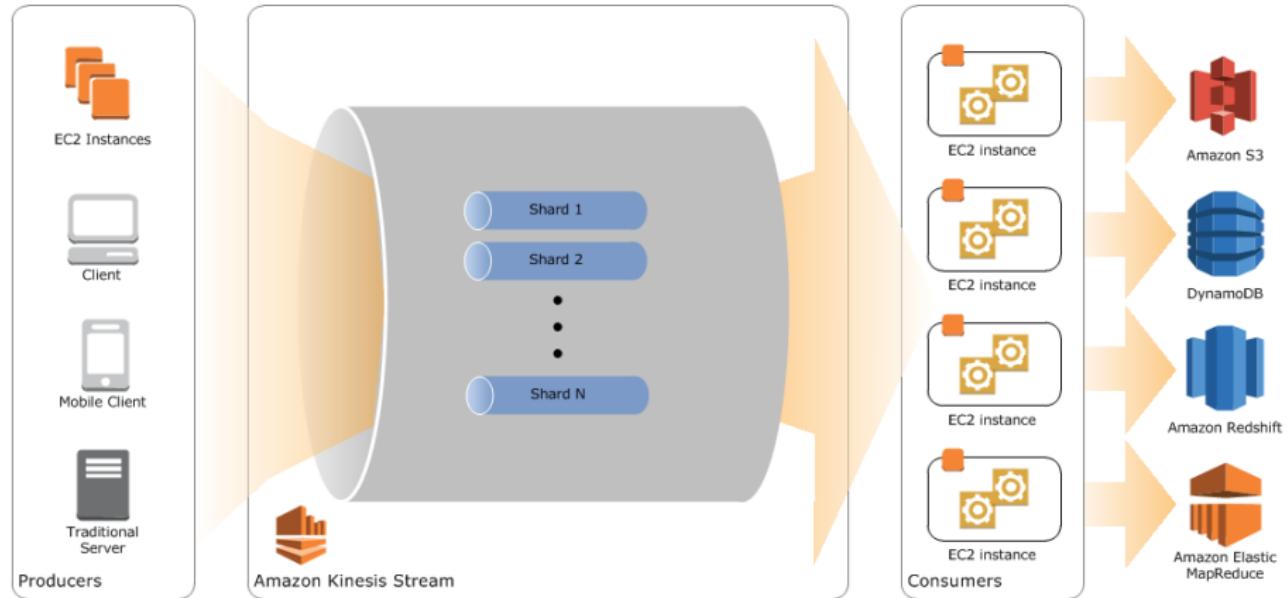
Source: SnowPlow

Why Amazon Kinesis & Redshift?



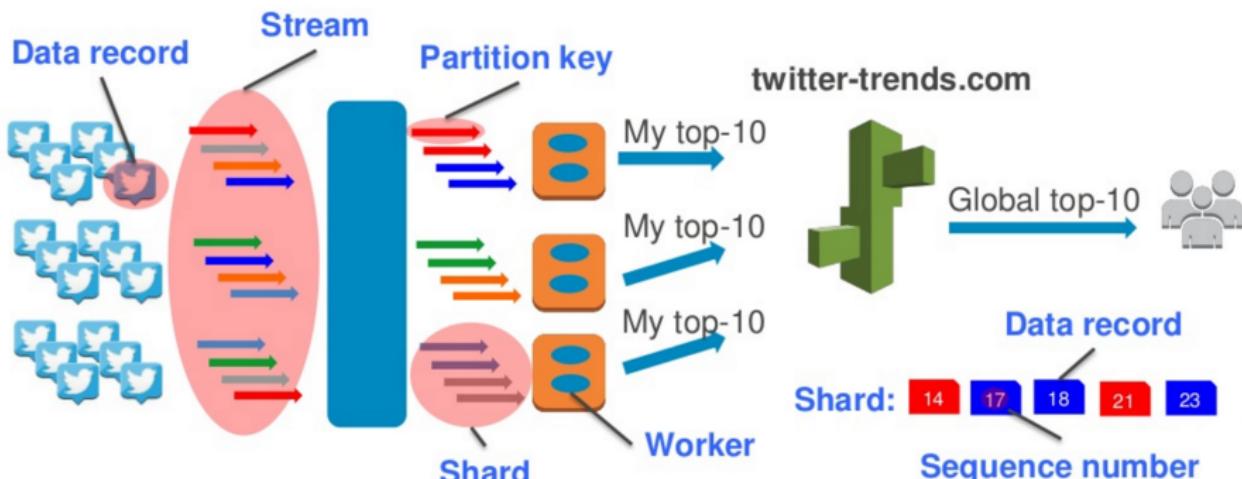
Source: Kinesis Product Details

Intro to Amazon Kinesis Streams



Source: Kinesis Developer Guide

Intro to Amazon Kinesis Shards



Source: AWS re:Invent 2013

Writing data to the stream:

- Amazon Kinesis Streams API (!)
- Amazon Kinesis Producer Library (KPL) from Java
- flume-kinesis
- Amazon Kinesis Agent

Reading data from the stream:

- Amazon Kinesis Streams API (!)
- Amazon Kinesis Client Library (KCL) from Java, Node.js, .NET, Python, Ruby

Managing streams:

- Amazon Kinesis Streams API (!)

Now We Need an R Client!

Initialize connection to the SDK:

```
> library(rJava)
+ .jinit(classpath =
+         list.files(
+             '~/Projects/kineric/inst/java/' ,
+             full.names = TRUE) ,
+         parameters = "-Xmx512m")
```

What methods do we have there?

```
> kc <- .jnew('com.amazonaws.services.kinesis.AmazonKinesisClient')
> .jmethods(kc)
[1] "public void com.amazonaws.services.kinesis.AmazonKinesisClient.splitShard(java"
[2] "public void com.amazonaws.services.kinesis.AmazonKinesisClient.splitShard(com"
[3] "public void com.amazonaws.services.kinesis.AmazonKinesisClient.removeTagsFrom"
[4] "public com.amazonaws.services.kinesis.model.ListStreamsResult com.amazonaws.s"
[5] "public com.amazonaws.services.kinesis.model.ListStreamsResult com.amazonaws.s"
[6] "public com.amazonaws.services.kinesis.model.ListStreamsResult com.amazonaws.s"
[7] "public com.amazonaws.services.kinesis.model.ListStreamsResult com.amazonaws.s"
[8] "public com.amazonaws.services.kinesis.model.GetShardIteratorResult com.amazon"
[9] "public com.amazonaws.services.kinesis.model.GetShardIteratorResult com.amazon"
[10] "public com.amazonaws.services.kinesis.model.GetShardIteratorResult com.amazon"
```

The First Steps from R

Set API endpoint:

```
> kc$setEndpoint('kinesis.us-west-2.amazonaws.com', 'kinesis', 'us-west-2')
```

What streams do we have access to?

```
> kc$listStreams()
[1] "Java-Object{{StreamNames: [test_kinesis],HasMoreStreams: false}}"
> kc$listStreams()$getStreamNames()
[1] "Java-Object{[test_kinesis]}"
```

Describe this stream:

```
> (kcstream <- kc$describeStream(StreamName = 'test_kinesis')$getStreamDescription()
[1] "Java-Object{{StreamName: test_kinesis,StreamARN: arn:aws:kinesis:us-west-2:595
> kcstream$getStreamName()
[1] "test_kinesis"
> kcstream$getStreamStatus()
[1] "ACTIVE"
> kcstream$getShards()
[1] "Java-Object{[
{ShardId: shardId-000000000000,HashKeyRange: {
StartingHashKey: 0,EndingHashKey: 17014118346046923173168730371588410572324},SequenceNumber: 17014118346046923173168730371588410572324}
]}"
```

Writing to the Stream

```
> prr <- .jnew('com.amazonaws.services.kinesis.model.PutRecordRequest')
> prr$setStreamName('test_kinesis')
> prr$setData(J('java.nio.ByteBuffer')$wrap(.jbyte(charToRaw('foobar'))))
> prr$setPartitionKey('42')
> prr
[1] "Java-Object{{StreamName: test_kinesis,
  Data: java.nio.HeapByteBuffer[pos=0 lim=6 cap=6],
  PartitionKey: 42,}}"
> kc$putRecord(prr)
[1] "Java-Object{{ShardId: shardId-000000000001,
  SequenceNumber: 49562894160449444332153346371084313572324361665031176210}}"
```

Get Records from the Stream

First, we need a shared iterator, which expires in 5 minutes:

```
> sir <- .jnew('com.amazonaws.services.kinesis.model.GetShardIteratorRequest')
> sir$setStreamName('test_kinesis')
> sir$setShardId(.jnew('java/lang/String', '0'))
> sir$setShardIteratorType('TRIM_HORIZON')
> iterator <- kc$getShardIterator(sir)$getShardIterator()
```

Then we can use it to get records:

```
> gir <- .jnew('com.amazonaws.services.kinesis.model.GetRecordsRequest')
> gir$setShardIterator(iterator)
> kc$getRecords(gir)$getRecords()
[1] "Java-Object{[]}"
```

Get Records from the Stream

First, we need a shared iterator, which expires in 5 minutes:

```
> sir <- .jnew('com.amazonaws.services.kinesis.model.GetShardIteratorRequest')
> sir$setStreamName('test_kinesis')
> sir$setShardId(.jnew('java/lang/String', '0'))
> sir$setShardIteratorType('TRIM_HORIZON')
> iterator <- kc$getShardIterator(sir)$getShardIterator()
```

Then we can use it to get records:

```
> gir <- .jnew('com.amazonaws.services.kinesis.model.GetRecordsRequest')
> gir$setShardIterator(iterator)
> kc$getRecords(gir)$getRecords()
[1] "Java-Object{[]}"
```

The data was pushed to the other shard:

```
> sir$setShardId(.jnew('java/lang/String', '1'))
> iterator <- kc$getShardIterator(sir)$getShardIterator()
> gir$setShardIterator(iterator)
> kc$getRecords(gir)$getRecords()
[1] "Java-Object[{SequenceNumber: 495628941604494443321533463710843135723243616650,
ApproximateArrivalTimestamp: Tue Jun 14 09:40:19 CEST 2016,
Data: java.nio.HeapByteBuffer[pos=0 lim=6 cap=6],
Records: 1}]"
```

Get Further Records from the Stream

```
> sapply(kc$getRecords(gir)$getRecords(),
+         function(x)
+           rawToChar(x$getData()$array()))
[1] "foobar"
> iterator <- kc$getRecords(gir)$getNextShardIterator()
> gir$setShardIterator(iterator)
> kc$getRecords(gir)$getRecords()
[1] "Java-Object{[]}"
```

Get the oldest available data again:

```
> iterator <- kc$getShardIterator(sir)$getShardIterator()
> gir$setShardIterator(iterator)
> kc$getRecords(gir)$getRecords()
[1] "Java-Object[{SequenceNumber: 495628941604494443321533463710843135723243616650}]
```

Or we can refer to the last known sequence number as well:

```
> sir$setShardIteratorType('AT_SEQUENCE_NUMBER')
> sir$setStartingSequenceNumber('495628941604494443321533463710843135723243616650311')
> iterator <- kc$getShardIterator(sir)$getShardIterator()
> gir$setShardIterator(iterator)
> kc$getRecords(gir)$getRecords()
[1] "Java-Object[{SequenceNumber: 495628941604494443321533463710843135723243616650}]
```

Managing Shards

No need for both shards:

```
> ms <- .jnew('com.amazonaws.services.kinesis.model.MergeShardsRequest')
> ms$setShardToMerge('shardId-000000000000')
> ms$setAdjacentShardToMerge('shardId-000000000001')
> ms$setStreamName('test_kinesis')
> kc$mergeShards(ms)
```

What do we have now?

```
> kc$describeStream(StreamName = 'test_kinesis')$getStreamDescription()$getShards()
[1] "Java-Object{[
{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey: 170
SequenceNumberRange: {
StartingSequenceNumber: 49562894160427143586954815717376297430913467927668719618,
EndingSequenceNumber: 49562894160438293959554081028945856364232263390243848194}},
{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey: 17014118346046923173
SequenceNumberRange: {
StartingSequenceNumber: 4956289416044944332153346340517833149186116289174700050,
EndingSequenceNumber: 49562894160460594704752611652087392082504911751749828626}},
{ShardId: shardId-000000000002,
ParentShardId: shardId-000000000000,
AdjacentParentShardId: shardId-000000000001,
HashKeyRange: {StartingHashKey: 0,EndingHashKey: 3402823669209384634633746074317682
SequenceNumberRange: {StartingSequenceNumber: 4956200499149767309970492434172701952}}
```

Next Steps

Ideas:

- R function managing/scaling shards and attaching R processors
- One R processor per shard (parallel threads on a node or cluster)
- Store started/finished sequence number in memory/DB (checkpointing)

Example (Design for a simple MVP on a single node)

mcparrallel starts parallel R processes to evaluate the given expression

Next Steps

Ideas:

- R function managing/scaling shards and attaching R processors
- One R processor per shard (parallel threads on a node or cluster)
- Store started/finished sequence number in memory/DB (checkpointing)

Example (Design for a simple MVP on a single node)

mcparrallel starts parallel R processes to evaluate the given expression

- or Gabor Csardi's `processx` package to manage child processes

Next Steps

Ideas:

- R function managing/scaling shards and attaching R processors
- One R processor per shard (parallel threads on a node or cluster)
- Store started/finished sequence number in memory/DB (checkpointing)

Example (Design for a simple MVP on a single node)

mcparrallel starts parallel R processes to evaluate the given expression

- or Gabor Csardi's `processx` package to manage child processes

Problems:

- duplicated records after failure due to in-memory checkpoint
- needs a DB for running on a cluster
- hard to write good unit tests

Next Steps

Ideas:

- R function managing/scaling shards and attaching R processors
- One R processor per shard (parallel threads on a node or cluster)
- Store started/finished sequence number in memory/DB (checkpointing)

Example (Design for a simple MVP on a single node)

mcparrallel starts parallel R processes to evaluate the given expression

- or Gabor Csardi's `processx` package to manage child processes

Problems:

- duplicated records after failure due to in-memory checkpoint
- needs a DB for running on a cluster
- hard to write good unit tests

Why not use KCL then?

- An easy-to-use programming model for processing data

```
java -cp amazon-kinesis-client-1.6.1.jar \
com.amazonaws.services.kinesis.multilang.MultiLangDaemon \
test-kinesis.properties
```

- Scale-out and fault-tolerant processing (checkpointing via DynamoDB)
- Logging and metrics in CloudWatch
- The MultiLangDaemon spawns processes written in any language, communication happens via JSON messages sent over stdin/stdout
- Only a few events/methods to care about in the consumer application:
 - ① initialize
 - ② processRecords
 - ③ checkpoint
 - ④ shutdown

① initialize:

- Perform initialization steps
- Write “status” message to indicate you are done
- Begin reading line from STDIN to receive next action

② processRecords:

- Perform processing tasks (you may write a checkpoint message at any time)
- Write “status” message to STDOUT to indicate you are done.
- Begin reading line from STDIN to receive next action

③ shutdown:

- Perform shutdown tasks (you may write a checkpoint message at any time)
- Write “status” message to STDOUT to indicate you are done.
- Begin reading line from STDIN to receive next action

④ checkpoint:

- Decide whether to checkpoint again based on whether there is an error or not.

Again: Why R?

User Defined Java Class

Step name: ShortenDate

Classes and code fragments:

- Classes
- Code Snippets
- Input fields
- Getting fields...please wait
- Info fields
- Getting fields...please wait
- Output fields
- Getting fields...please wait

Class code

```

# Processor
import java.text.SimpleDateFormat;
import java.util.Date;
import java.text.ParseException;
import java.util.TimeZone;

private SimpleDateFormat df1 = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss,SSS");

private SimpleDateFormat df2 = new SimpleDateFormat("yyyy-MM-dd HH");

public boolean processRow(StepMetaInterface smi, StepDataInterface sdi) throws KettleException, ParseException
{
    Object[] r = getRow();
    if (r == null) {
        setOutputDone();
        return false;
    }

    if (first)
    {
        first = false;
    }

    // It is always safest to call createOutputRow() to ensure that your output row's Object[] is large
    // enough to handle any new fields you are creating in this step.
    r = createOutputRow(r, data.outputRowMeta.size());

    df2.setTimeZone(TimeZone.getTimeZone("America/Los_Angeles"));

    df2.set

```

Line #: 0

Fields | Parameters | Info steps | Target steps

Fields

#	Fieldname	Type	Length	Precision
1	RPT_DATE_SHORT	String		

Clear the result fields?

Help OK Cancel Test class

R Script Interacting with KCL

```
#!/usr/bin/r -i

while (TRUE) {

    ## read and parse messages
    line <- fromJSON(readLines(n = 1))

    ## nothing to do unless we receive a record to process
    if (line$action == 'processRecords') {

        ## process each record
        lapply(line$records, function(r) {

            business_logic(fromJSON(rawToChar(base64_dec(r$data))))
            catn(toJSON(list(action = 'checkpoint', checkpoint = r$sequenceNumber)))

        })
    }

    ## return response in JSON
    catn(toJSON(list(action = 'status', responseFor = line$action)))

}
```

Examples

