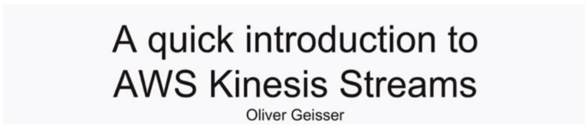
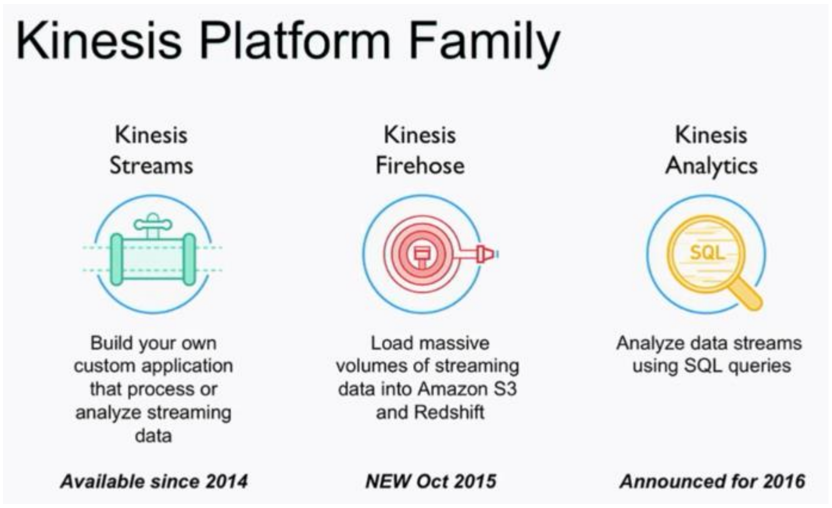


A quick introduction to AWS Kinesis Streams

Oliver Geisser



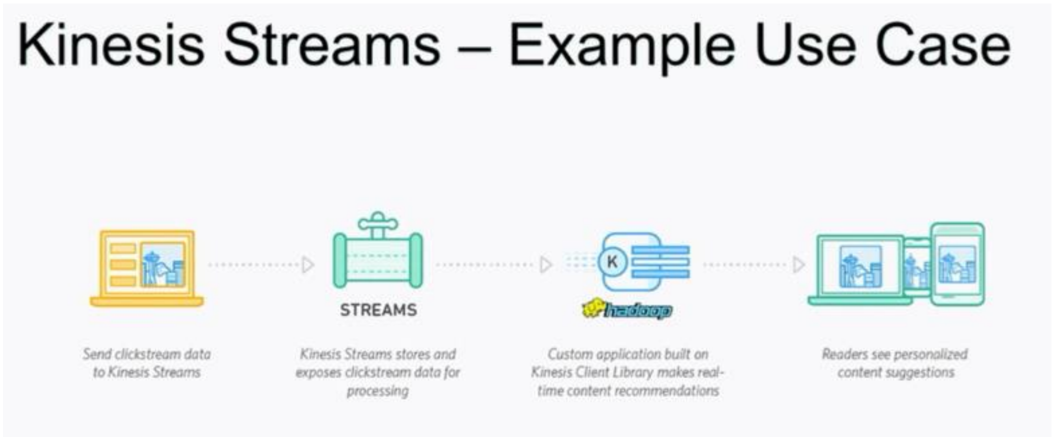
Kinesis Platform Family

Kinesis Streams	Kinesis Firehose	Kinesis Analytics
Build your own custom application that process or analyze streaming data	Load massive volumes of streaming data into Amazon S3 and Redshift	Analyze data streams using SQL queries
Available since 2014	***NEW Oct 2015***	***Announced for 2016***

Kinesis Platform Family

Kinesis Streams	Kinesis Firehose	Kinesis Analytics
Build your own custom application that process or analyze streaming Data	Load massive volumes of streaming data into Amazon S3 and Redshift	Analyze data streams using SQL queries
Available since 2014	***NEW Oct 2015***	***Announced for 2016***

Kinesis Streams is like a managed Kafka cluster that allows you store event data into a place before analyzing them



Kinesis Streams – Example Use Case

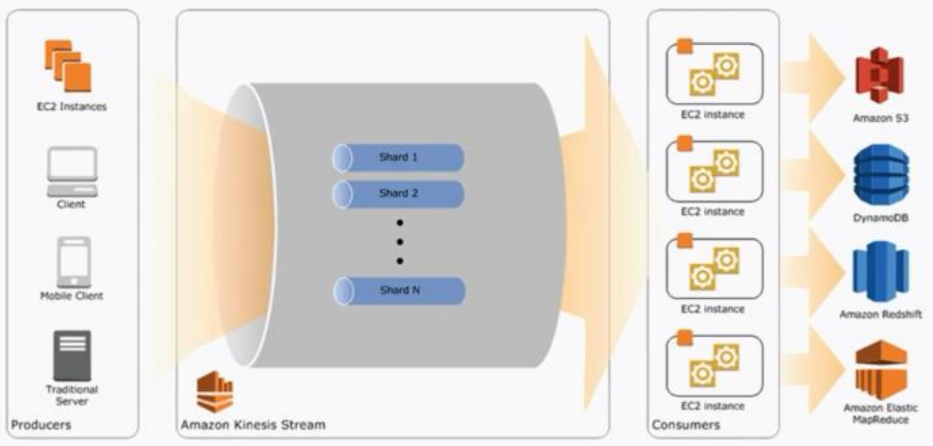
Send clickstream data to Kinesis Streams

Kinesis Streams stores and exposes clickstream data for processing

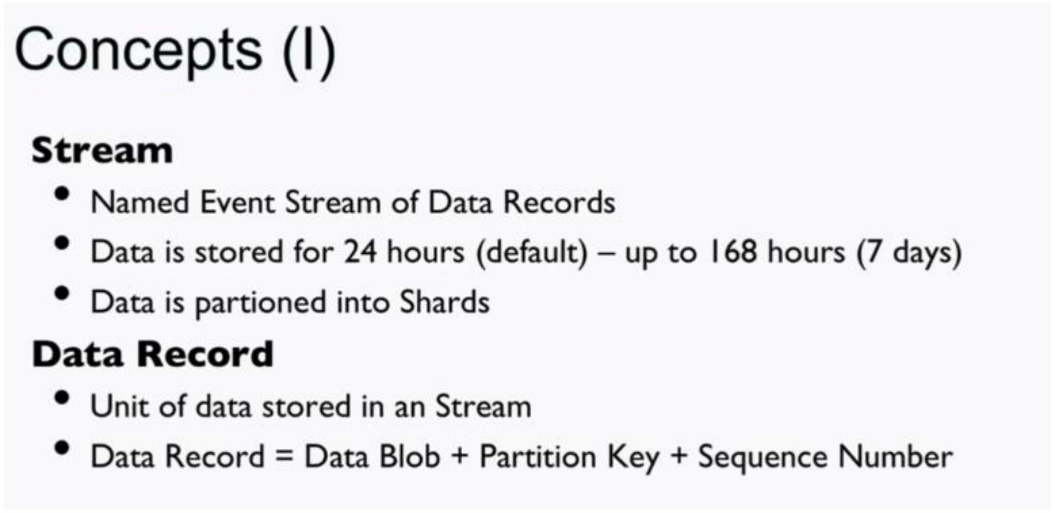
Custom application built on Kinesis Client Library makes real-time content recommendations

Readers see personalized content suggestions

High Level Architecture



The data is put in Streams, the Streams is divided into Shards, Consumers can retrieve records from the shards and do something with them.



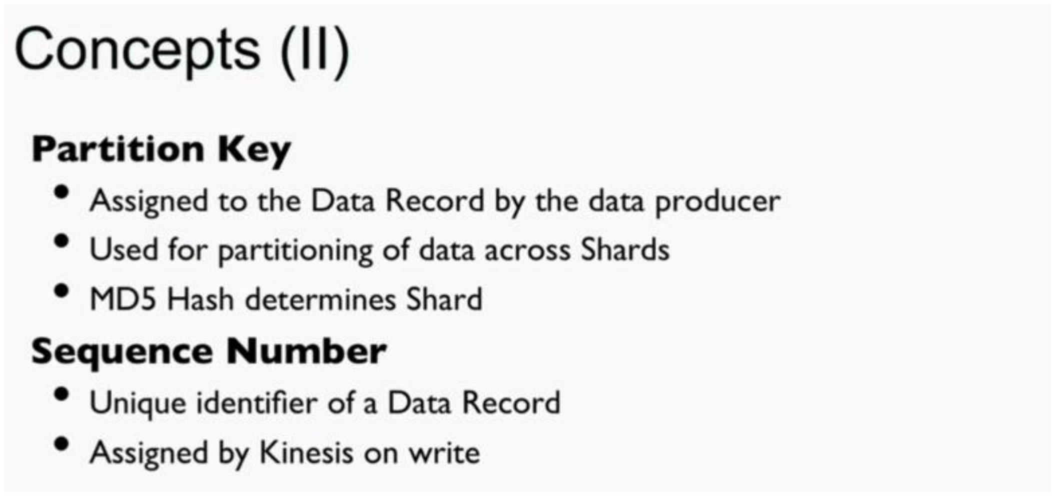
Concepts (I)

Stream

- Named Event Stream of Data Records
- Data is stored for 24 hours (default) – up to 168 hours (7 days)
- Data is partioned into Shards

Data Record

- Unit of data stored in an Stream
- Data Record = Data Blob + Partition Key + Sequence Number



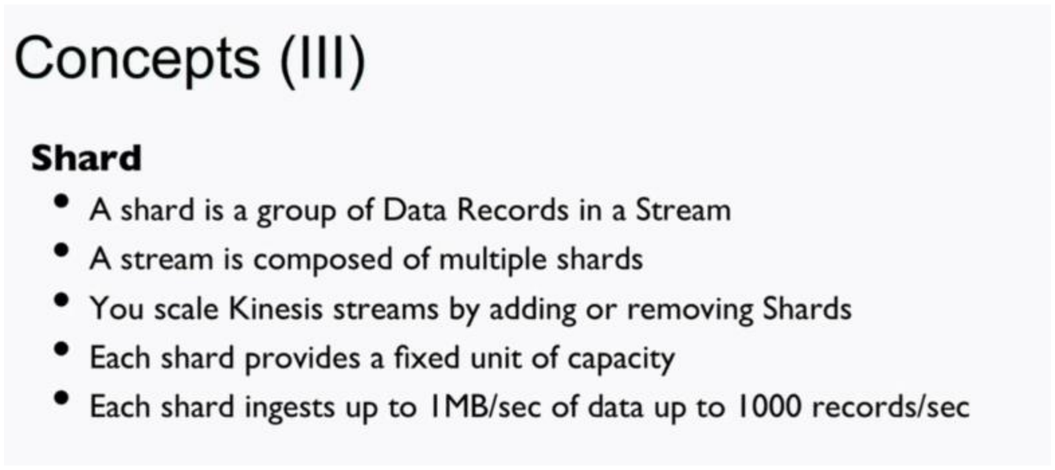
Concepts (II)

Partition Key

- Assigned to the Data Record by the data producer
- Used for partitioning of data across Shards
- MD5 Hash determines Shard

Sequence Number

- Unique identifier of a Data Record
- Assigned by Kinesis on write

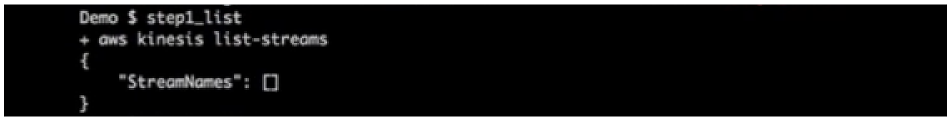


Concepts (III)

Shard

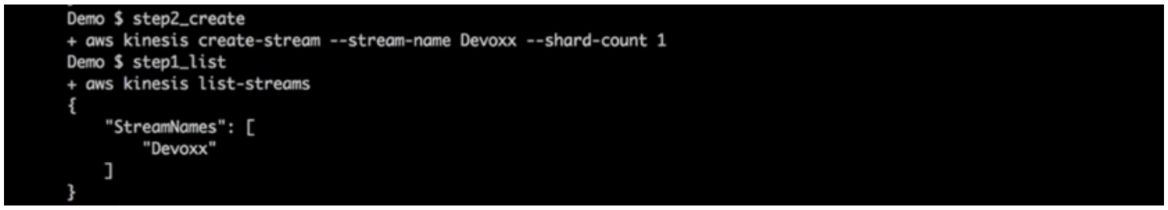
- A shard is a group of Data Records in a Stream
- A stream is composed of multiple shards
- You scale Kinesis streams by adding or removing Shards
- Each shard provides a fixed unit of capacity
- Each shard ingests up to 1MB/sec of data up to 1000 records/sec

Demo



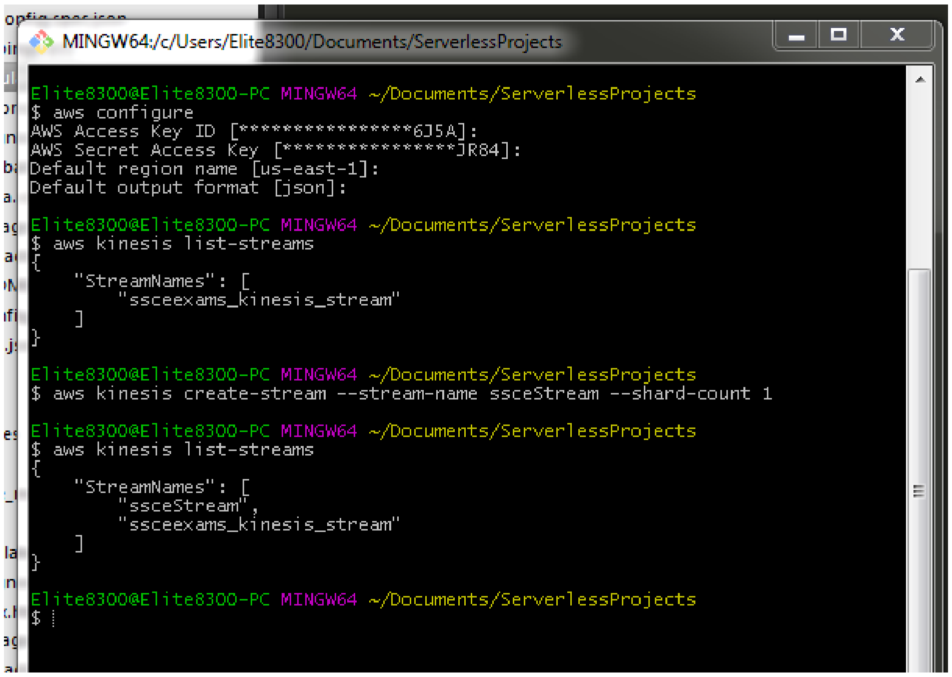
```
Demo $ step1_list
+ aws kinesis list-streams
{
    "StreamNames": []
}
```

Using the AWS CLI, you can list your available streams using the ***$ aws kinesis list-streams*** command

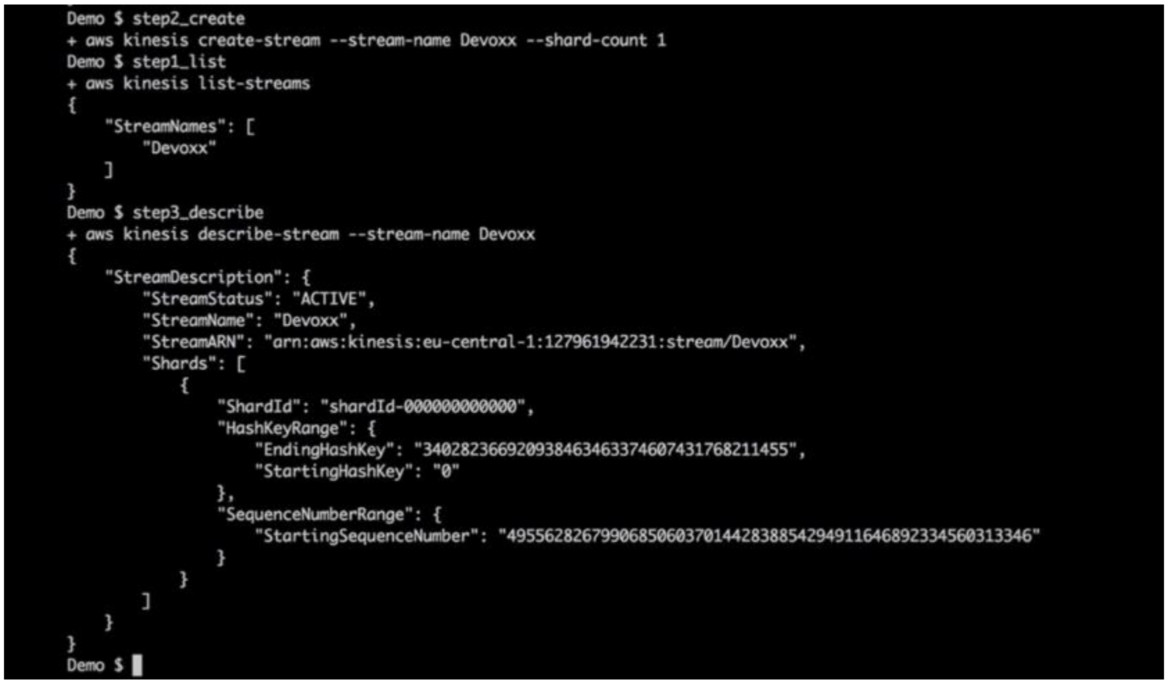


```
Demo $ step2_create
+ aws kinesis create-stream --stream-name Devoxx --shard-count 1
Demo $ step1_list
+ aws kinesis list-streams
{
    "StreamNames": [
        "Devoxx"
    ]
}
```

We can create a new stream with command ***$ aws kinesis create-stream - -stream-name ssceStream - -shard-count 1.*** we can increase or decrease the number of shards later.

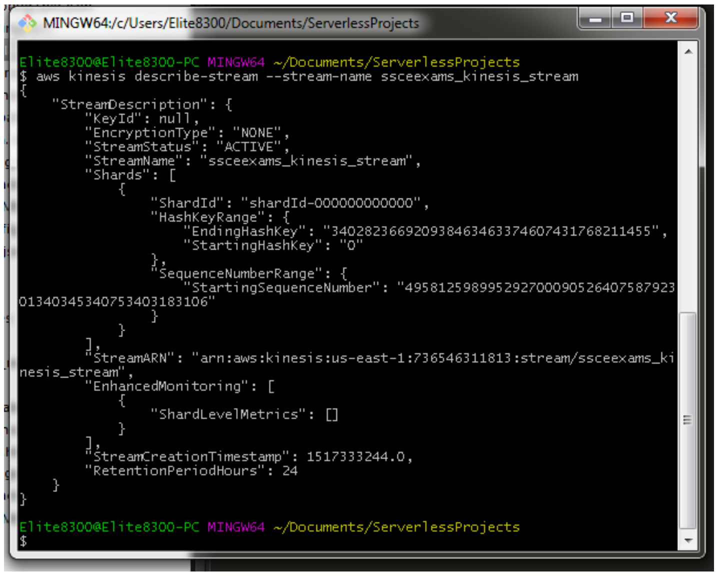


```
MINGW64:/c/Users/Elite8300/Documents/ServerlessProjects

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws configure
AWS Access Key ID [****************6J5A]:
AWS Secret Access Key [****************JR84]:
Default region name [us-east-1]:
Default output format [json]:

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis list-streams
{
    "StreamNames": [
        "ssceexams_kinesis_stream"
    ]
}

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis create-stream --stream-name ssceStream --shard-count 1

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis list-streams
{
    "StreamNames": [
        "ssceStream",
        "ssceexams_kinesis_stream"
    ]
}

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$
```

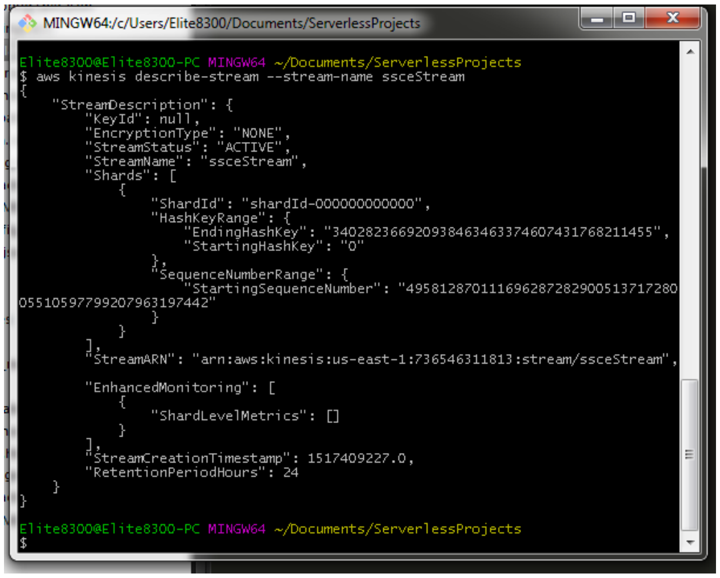


```
Demo $ step2_create
+ aws kinesis create-stream --stream-name Devoxx --shard-count 1
Demo $ step1_list
+ aws kinesis list-streams
{
    "StreamNames": [
        "Devoxx"
    ]
}
Demo $ step3_describe
+ aws kinesis describe-stream --stream-name Devoxx
{
    "StreamDescription": {
        "StreamStatus": "ACTIVE",
        "StreamName": "Devoxx",
        "StreamARN": "arn:aws:kinesis:eu-central-1:127961942231:stream/Devoxx",
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "EndingHashKey": "340282366920938463463374607431768211455",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber": "49556282679906850603701442838854294911646892334560313346"
                }
            }
        ]
    }
}
Demo $ 
```

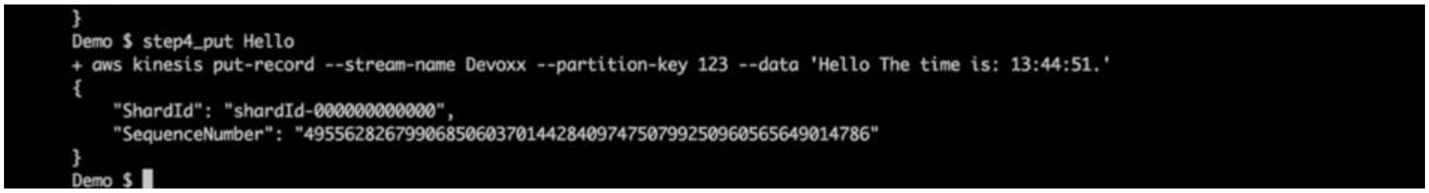
We can use the ***$ aws kinesis describe-stream - -stream-name Devoxx*** to see details about the kinesis stream as above. We can see that the stream is ACTIVE, has 1 shard with a ShardID, HashKeyRange and a SequenceNumberRange. All partition keys will go into the sequence value.



```
Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis describe-stream --stream-name ssceexams_kinesis_stream
{
    "StreamDescription": {
        "KeyId": null,
        "EncryptionType": "NONE",
        "StreamStatus": "ACTIVE",
        "StreamName": "ssceexams_kinesis_stream",
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "EndingHashKey": "340282366920938463463374607431768211455",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber": "4958125989952927000905264075879230134034534075340318310 6"
                }
            }
        ],
        "StreamARN": "arn:aws:kinesis:us-east-1:736546311813:stream/ssceexams_kinesis_stream",
        "EnhancedMonitoring": [
            {
                "ShardLevelMetrics": []
            }
        ],
        "StreamCreationTimestamp": 1517333244.0,
        "RetentionPeriodHours": 24
    }
}

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$
```

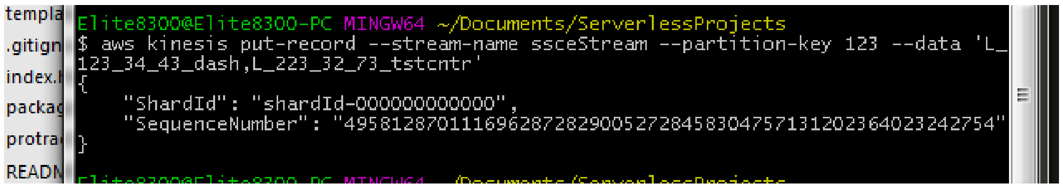


```
Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis describe-stream --stream-name ssceStream
{
    "StreamDescription": {
        "KeyId": null,
        "EncryptionType": "NONE",
        "StreamStatus": "ACTIVE",
        "StreamName": "ssceStream",
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "EndingHashKey": "340282366920938463463374607431768211455",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber": "49581287011169628728290051371728005510597799207963197442"
                }
            }
        ],
        "StreamARN": "arn:aws:kinesis:us-east-1:736546311813:stream/ssceStream",

        "EnhancedMonitoring": [
            {
                "ShardLevelMetrics": []
            }
        ],
        "StreamCreationTimestamp": 1517409227.0,
        "RetentionPeriodHours": 24
    }
}

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$
```



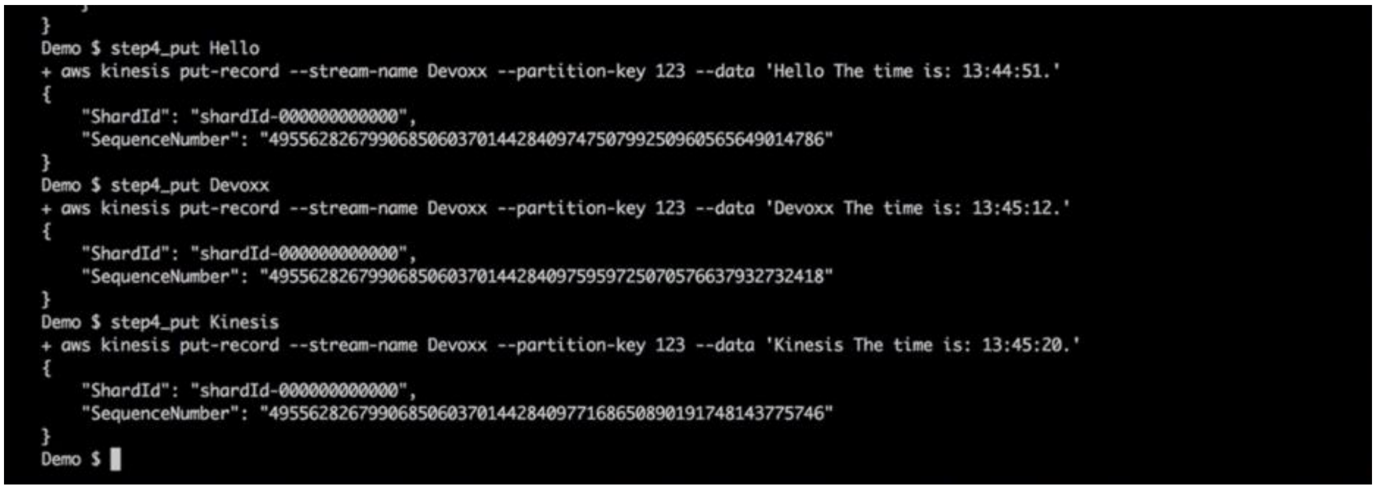
```
}
Demo $ step4_put Hello
+ aws kinesis put-record --stream-name Devoxx --partition-key 123 --data 'Hello The time is: 13:44:51.'
{
    "ShardId": "shardId-000000000000",
    "SequenceNumber": "49556282679906850603701442840974750799250960565649014786"
}
Demo $ 
```

We can put some data into the Devoxx Kinesis stream using the command ***$ aws kinesis put-record - -stream-name Devoxx - -partition-key 123 - -data 'Hello The time is 13:44:51.'***



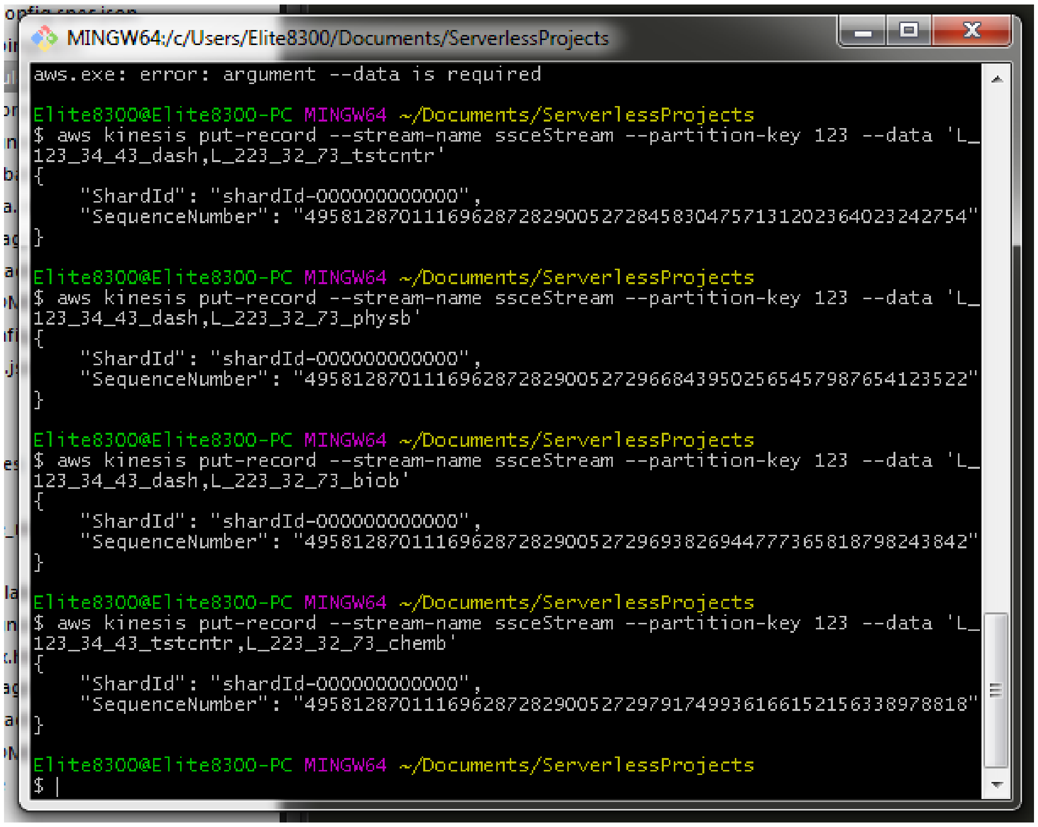
```
Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis put-record --stream-name ssceStream --partition-key 123 --data 'L_123_34_43_dash,L_223_32_73_tstcntr'
{
    "ShardId": "shardId-000000000000",
    "SequenceNumber": "49581287011169628728290052728458304757131202364023242754"
}

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
```

You can put up to 1MB data into the stream at once

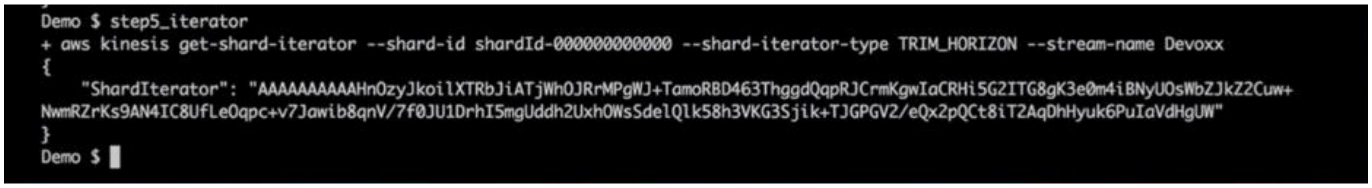


```
}
Demo $ step4_put Hello
+ aws kinesis put-record --stream-name Devoxx --partition-key 123 --data 'Hello The time is: 13:44:51.'
{
    "ShardId": "shardId-000000000000",
    "SequenceNumber": "49556282679906850603701442840974750799250960565649014786"
}
Demo $ step4_put Devoxx
+ aws kinesis put-record --stream-name Devoxx --partition-key 123 --data 'Devoxx The time is: 13:45:12.'
{
    "ShardId": "shardId-000000000000",
    "SequenceNumber": "49556282679906850603701442840975959725070576637932732418"
}
Demo $ step4_put Kinesis
+ aws kinesis put-record --stream-name Devoxx --partition-key 123 --data 'Kinesis The time is: 13:45:20.'
{
    "ShardId": "shardId-000000000000",
    "SequenceNumber": "49556282679906850603701442840977168650890191748143775746"
}
Demo $ 
```

We now have 3 data records in the Kinesis stream as above. Note that the response is always the shard (with the ShardID and SequenceNumber) that the record was inserted into.

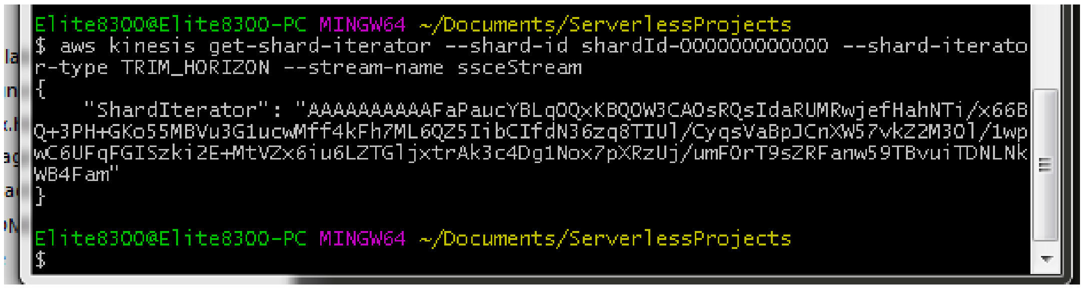


```
MINGW64:/c/Users/Elite8300/Documents/ServerlessProjects
aws.exe: error: argument --data is required

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis put-record --stream-name ssceStream --partition-key 123 --data 'L_
123_34_43_dash,L_223_32_73_tstcntr'
{
    "ShardId": "shardId-000000000000",
    "SequenceNumber": "49581287011169628728290052728458304757131202364023242754"
}

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis put-record --stream-name ssceStream --partition-key 123 --data 'L_
123_34_43_dash,L_223_32_73_physb'
{
    "ShardId": "shardId-000000000000",
    "SequenceNumber": "49581287011169628728290052729668439502565457987654123522"
}

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis put-record --stream-name ssceStream --partition-key 123 --data 'L_
123_34_43_dash,L_223_32_73_biob'
{
    "ShardId": "shardId-000000000000",
    "SequenceNumber": "49581287011169628728290052729693826944777365818798243842"
}

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis put-record --stream-name ssceStream --partition-key 123 --data 'L_
123_34_43_tstcntr,L_223_32_73_chemb'
{
    "ShardId": "shardId-000000000000",
    "SequenceNumber": "49581287011169628728290052729791749936166152156338978818"
}

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ |
```



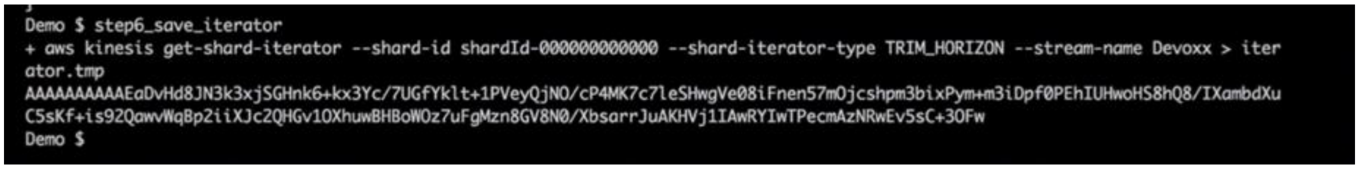
```
Demo $ step5_iterator
+ aws kinesis get-shard-iterator --shard-id shardId-000000000000 --shard-iterator-type TRIM_HORIZON --stream-name Devoxx
{
    "ShardIterator": "AAAAAAAAAAHnOzyJkoilXTRbJiATjWhOJRrMPgWJ+TamoRBD463ThggdQqpRJCrmKgwIaCRHi5G2ITG8gK3e0m4iBNyUOsWbZJkZ2Cuw+
NwmRZrKs9AN4IC8UfLeOqpc+v7Jawib8qnV/7f0JU1DrhI5mgUddh2UxhOWsSdelQlk58h3VKG3Sjik+TJGPGV2/eQx2pQCt8iT2AqDhHyuk6PuIaVdHgUW"
}
Demo $ 
```

We can retrieve data back from the Kinesis stream using the Iterator, which is like a temporary cursor using the command ***$ aws kinesis get-shard-iterator - -shard-id shardId-000000000 - -shard-iterator-type TRIM_HORIZON - -stream-name Devoxx***



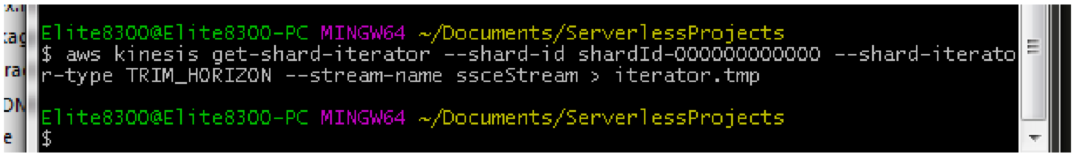
```
Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis get-shard-iterator --shard-id shardId-000000000000 --shard-iterato
r-type TRIM_HORIZON --stream-name ssceStream
{
    "ShardIterator": "AAAAAAAAAAFaPaucYBLqOQxKBQOW3CAOsRQsIdaRUMRwjefHahNTi/x66B
Q+3PH+GKo55MBVu3G1ucwMff4kFh7ML6QZ5IibCIfdN36zq8TIUl/CyqsVaBpJCnXW57vkZ2M3Ol/1wp
wC6UFqFGISzki2E+MtVZx6iu6LZTGljxtrAk3c4Dg1Nox7pXRzUj/umFOrT9sZRFanw59TBvuiTDNLNk
WB4Fam"
}

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$
```

The lifetime of the iterator is limited to 5 minutes, so you always need to get a new iterator for retrieving the data from the stream.

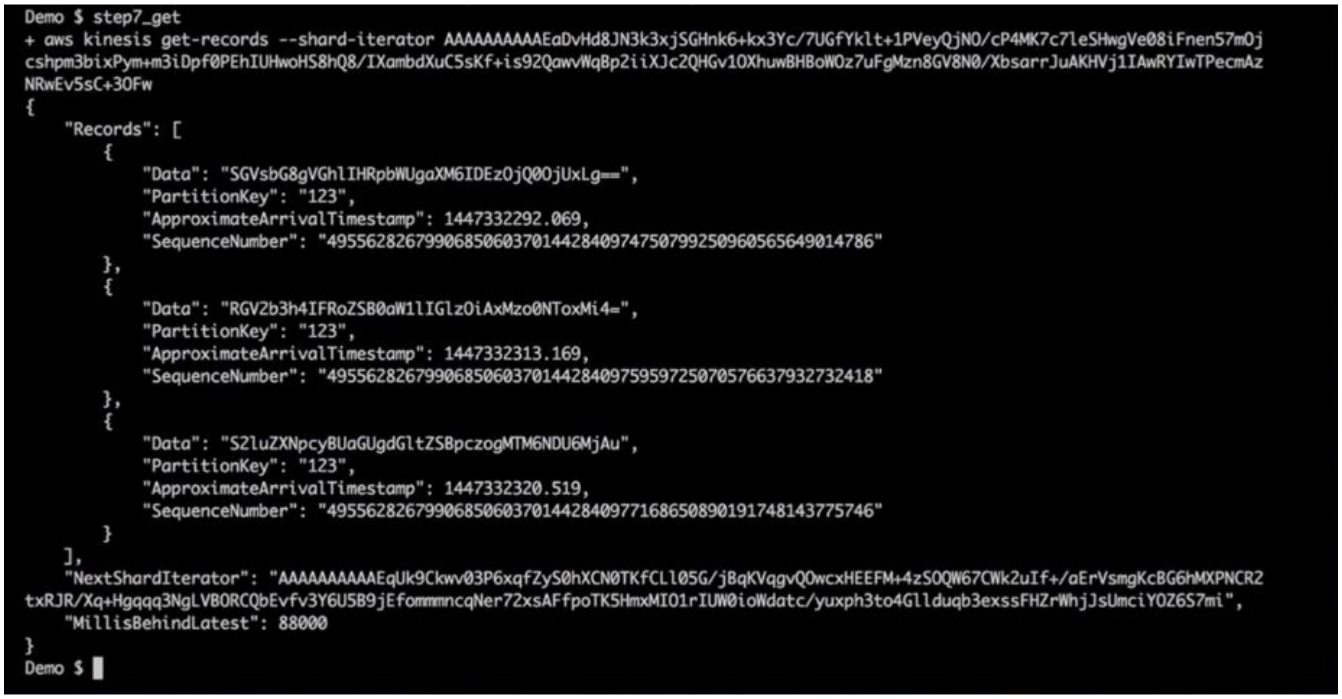


```
Demo $ step6_save_iterator
+ aws kinesis get-shard-iterator --shard-id shardId-000000000000 --shard-iterator-type TRIM_HORIZON --stream-name Devoxx > iter
ator.tmp
AAAAAAAAAAEaDvHd8JN3k3xjSGHnk6+kx3Yc/7UGfYklt+1PVeyQjNO/cP4MK7c7leSHwgVe08iFnen57mOjcshpm3bixPym+m3iDpf0PEhIUHwoHS8hQ8/IXambdXu
C5sKf+is92QawvWqBp2iiXJc2QHGv1OXhuwBHBoWOz7uFgMzn8GV8N0/XbsarrJuAKHVj1IAwRYIwTPecmAzNRwEv5sC+3OFw
Demo $
```

We can save this iterator in a temp file so that we can use it for a while before getting a new one. We use the command ***$ aws kinesis get-shard-iterator - -shard-id shardID-000000000000 - -shard-iterator-type TRIM_HORIZON - -stream-name Devoxx > iterator.tmp***

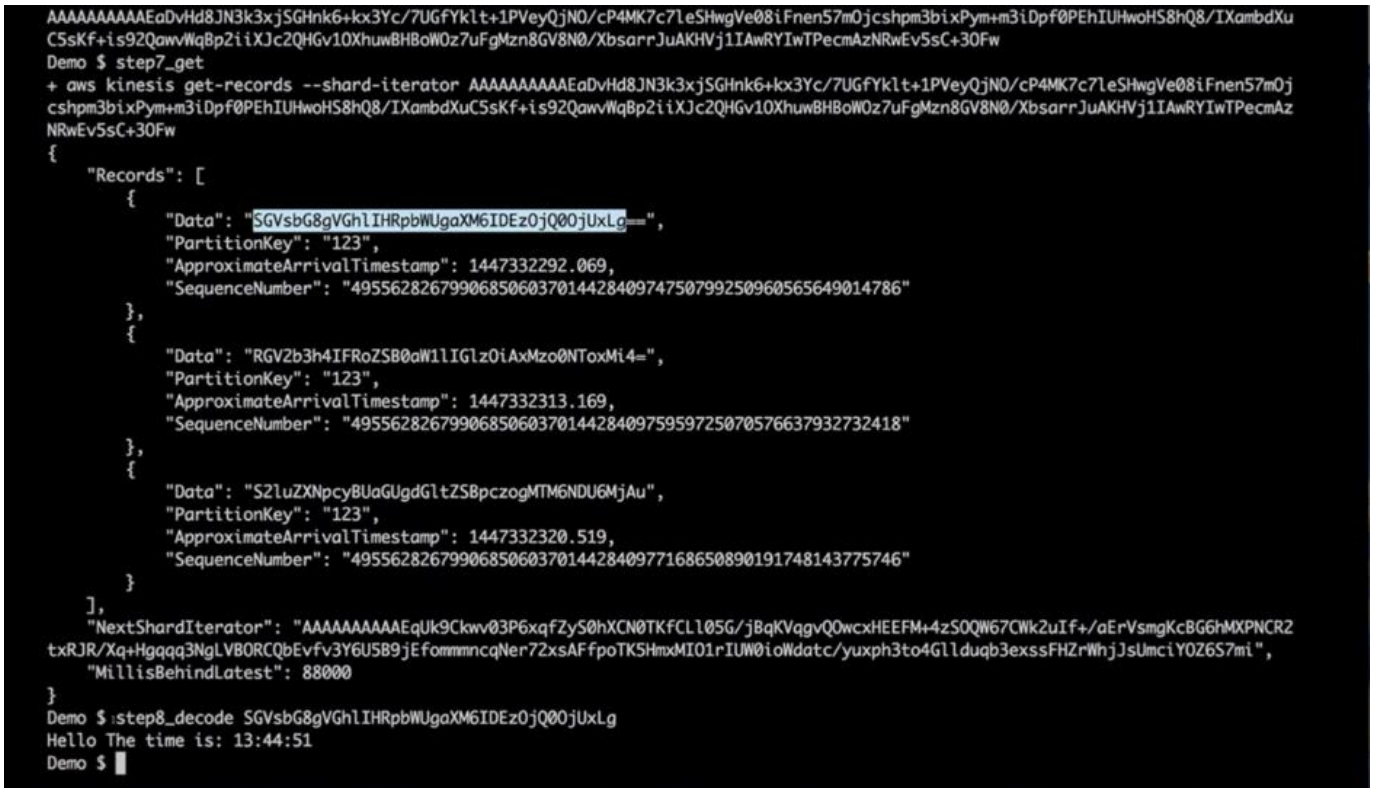


```
Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis get-shard-iterator --shard-id shardId-000000000000 --shard-iterato
r-type TRIM_HORIZON --stream-name ssceStream > iterator.tmp

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$
```



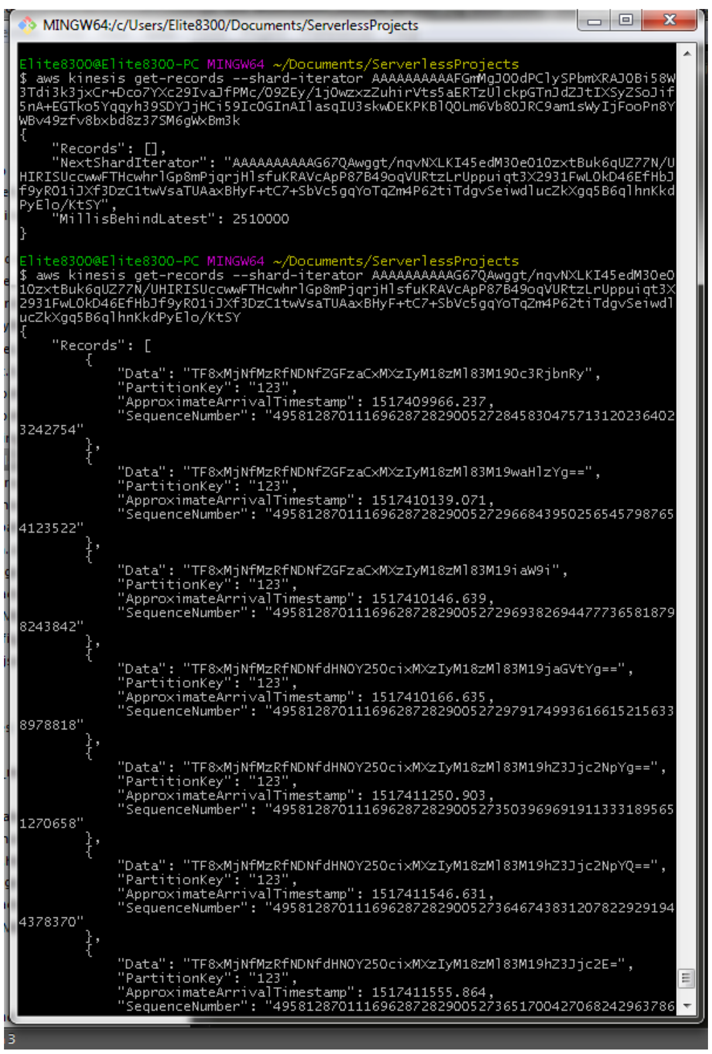
```
Demo $ step7_get
+ aws kinesis get-records --shard-iterator AAAAAAAAAAEaDvHd8JN3k3xjSGHnk6+kx3Yc/7UGfYklt+1PVeyQjNO/cP4MK7c7leSHwgVe08iFnen57mOj
cshpm3bixPym+m3iDpf0PEhIUHwoHS8hQ8/IXambdXuC5sKf+is92QawvWqBp2iiXJc2QHGv1OXhuwBHBoWOz7uFgMzn8GV8N0/XbsarrJuAKHVj1IAwRYIwTPecmAz
NRwEv5sC+3OFw
{
    "Records": [
        {
            "Data": "SGVsbG8gVGhlIHRpbWUgaXM6IDEzOjQ0OjUxLg==",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1447332292.069,
            "SequenceNumber": "49556282679906850603701442840974750799250960565649014786"
        },
        {
            "Data": "RGV2b3h4IFRoZSB0aW1lIGlzOiAxMzo0NToxMi4=",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1447332313.169,
            "SequenceNumber": "49556282679906850603701442840975959725070576637932732418"
        },
        {
            "Data": "S2luZXNpcyBUaGUgdGltZSBpczogMTM6NDU6MjAu",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1447332320.519,
            "SequenceNumber": "49556282679906850603701442840977168650890191748143775746"
        }
    ],
    "NextShardIterator": "AAAAAAAAAAEqUk9Ckwv03P6xqfZyS0hXCN0TKfCLl05G/jBqKVqgvQOwcxHEEFM+4zSOQW67CWk2uIf+/aErVsmgKcBG6hMXPNCR2
txRJR/Xq+Hgqqq3NgLVBORCQbEvfv3Y6U5B9jEfommncqNer72xsAFfpoTK5HmxMIO1rIUW0ioWdatc/yuxph3to4Gllduqb3exssFHZrWhjJsUmciYOZ6S7mi",
    "MillisBehindLatest": 88000
}
Demo $ █
```

We can now get the data records using the currently saved iterator using the command ***$ aws kinesis get-records - -shard-iterator***
AAAAAAAAAAFaPaucYBLqOQxKBQOW3CAOsRQsIdaRUMRwjefHahNTi/x66BQ+3PH+GKo55MBVu3G1ucwMff4kFh7ML6QZ5IibCIfdN36zq8TIUl/CyqsVaBpJCnXW57vkZ2M3Ol/1wpwC6UFqFGISzki2E+MtVZx6iu6LZTGljxtrAk3c4Dg1Nox7pXRzUj/umFOrT9sZRFanw59TBvuiTDNLNkWB4Fam

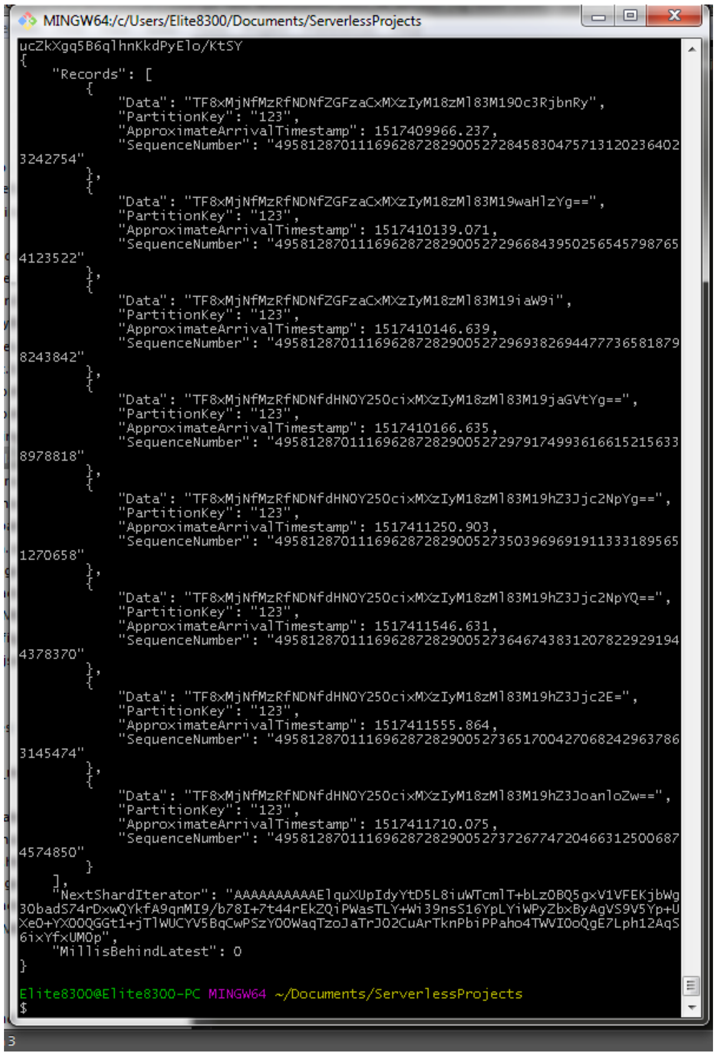


```
AAAAAAAAAAEaDvHd8JN3k3xjSGHnk6+kx3Yc/7UGfYklt+1PVeyQjNO/cP4MK7c7leSHwgVe08iFnen57mOjcshpm3bixPym+m3iDpf0PEhIUHwoHS8hQ8/IXambdXu
C5sKf+is92QawvWqBp2iiXJc2QHGv1OXhuwBHBoWOz7uFgMzn8GV8N0/XbsarrJuAKHVj1IAwRYIwTPecmAzNRwEv5sC+3OFw
Demo $ step7_get
+ aws kinesis get-records --shard-iterator AAAAAAAAAAEaDvHd8JN3k3xjSGHnk6+kx3Yc/7UGfYklt+1PVeyQjNO/cP4MK7c7leSHwgVe08iFnen57mOj
cshpm3bixPym+m3iDpf0PEhIUHwoHS8hQ8/IXambdXuC5sKf+is92QawvWqBp2iiXJc2QHGv1OXhuwBHBoWOz7uFgMzn8GV8N0/XbsarrJuAKHVj1IAwRYIwTPecmAz
NRwEv5sC+3OFw
{
    "Records": [
        {
            "Data": "SGVsbG8gVGhlIHRpbWUgaXM6IDEzOjQ0OjUxLg==",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1447332292.069,
            "SequenceNumber": "49556282679906850603701442840974750799250960565649014786"
        },
        {
            "Data": "RGV2b3h4IFRoZSB0aW1lIGlzOiAxMzo0NToxMi4=",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1447332313.169,
            "SequenceNumber": "49556282679906850603701442840975959725070576637932732418"
        },
        {
            "Data": "S2luZXNpcyBUaGUgdGltZSBpczogMTM6NDU6MjAu",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1447332320.519,
            "SequenceNumber": "49556282679906850603701442840977168650890191748143775746"
        }
    ],
    "NextShardIterator": "AAAAAAAAAAEqUk9Ckwv03P6xqfZyS0hXCN0TKfCLl05G/jBqKVqgvQOwcxHEEFM+4zSOQW67CWk2uIf+/aErVsmgKcBG6hMXPNCR2
txRJR/Xq+Hgqqq3NgLVBORCQbEvfv3Y6U5B9jEfommncqNer72xsAFfpoTK5HmxMIO1rIUW0ioWdatc/yuxph3to4Glld uqb3exssFHZrWhjJsUmciYOZ6S7mi",
    "MillisBehindLatest": 88000
}
Demo $ step8_decode SGVsbG8gVGhlIHRpbWUgaXM6IDEzOjQ0OjUxLg
Hello The time is: 13:44:51
Demo $ █
```

We have 3 data records from the Kinesis stream, the Data has been Base64 encoded and we need to decode the values using the decode command ***$ decode sgdgifgoghkglhlihlholihol***

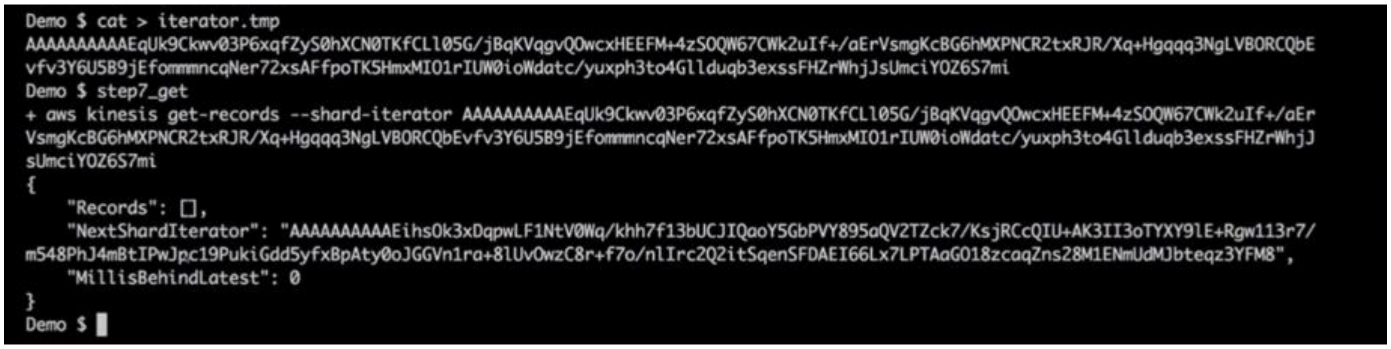


```
Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis get-records --shard-iterator AAAAAAAAAAFGmMgJOOdPClySPbmXRAJOBi58W
3Tdi3k3jxCr+Dco7YXc29IvaJfPMc/O9ZEy/1jOwzxzZuhirVts5aERTzUlckpGTnJdZJtIXSyZSoJif
5nA+EGTko5Yqqyh39SDYJjHCi59IcOGInAIlasqIU3skwDEKPKBlQOLm6Vb8OJRC9am1sWyIjFooPn8Y
WBv49zfv8bxbd8z37SM6gWxBm3k
{
    "Records": [],
    "NextShardIterator": "AAAAAAAAAAG67QAwggt/nqvNXLKI45edM3Oe01OzxtBuk6qUZ77N/U
HIRISUccwwFTHcwhrlGp8mPjqrjHlsfuKRAVcApP87B49oqVURtzLrUppuiqt3X2931FwLOkD46EfHbJ
f9yRO1iJXf3DzC1twVsaTUAaxBHyF+tC7+SbVc5gqYoTqZm4P62tiTdgvSeiwdlucZkXgq5B6qlhnKkd
PyElo/KtSY",
    "MillisBehindLatest": 2510000
}

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$ aws kinesis get-records --shard-iterator AAAAAAAAAAG67QAwggt/nqvNXLKI45edM3OeO
1OzxtBuk6qUZ77N/UHIRISUccwwFTHcwhrlGp8mPjqrjHlsfuKRAVcApP87B49oqVURtzLrUppuiqt3X
2931FwLOkD46EfHbJf9yRO1iJXf3DzC1twVsaTUAaxBHyF+tC7+SbVc5gqYoTqZm4P62tiTdgvSeiwdl
ucZkXgq5B6qlhnKkdPyElo/KtSY
{
    "Records": [
        {
            "Data": "TF8xMjNfMzRfNDNfZGFzaCxMXzIyM18zMl83M19Oc3RjbnRy",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517409966.237,
            "SequenceNumber": "4958128701116962872829005272845830475713120236402
3242754"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfZGFzaCxMXzIyM18zMl83M19waHlzYg==",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517410139.071,
            "SequenceNumber": "4958128701116962872829005272966843950256545798765
4123522"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfZGFzaCxMXzIyM18zMl83M19iaW9i",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517410146.639,
            "SequenceNumber": "4958128701116962872829005272969382694477736581879
8243842"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfdHNOY25OcixMXzIyM18zMl83M19jaGVtYg==",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517410166.635,
            "SequenceNumber": "4958128701116962872829005272979174993616615215633
8978818"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfdHNOY25OcixMXzIyM18zMl83M19hZ3Jjc2NpYg==",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517411250.903,
            "SequenceNumber": "4958128701116962872829005273503969691911333189565
1270658"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfdHNOY25OcixMXzIyM18zMl83M19hZ3Jjc2NpYQ==",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517411546.631,
            "SequenceNumber": "4958128701116962872829005273646743831207822929194
4378370"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfdHNOY25OcixMXzIyM18zMl83M19hZ3Jjc2E=",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517411555.864,
            "SequenceNumber": "4958128701116962872829005273651700427068242963786
```



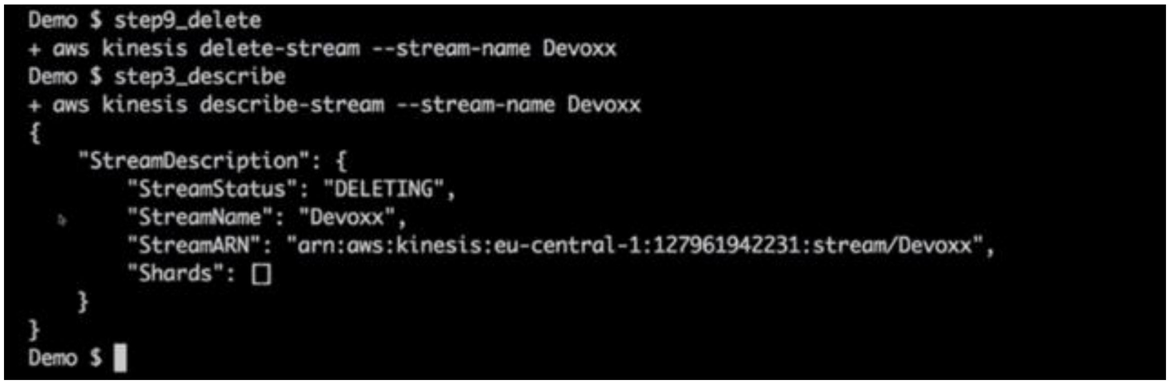
```
ucZkXgq5B6qlhnKkdPyElo/KtSY
{
    "Records": [
        {
            "Data": "TF8xMjNfMzRfNDNfZGFzaCxMXzIyM18zMl83M19Oc3RjbnRy",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517409966.237,
            "SequenceNumber": "4958128701116962872829005272845830475713120236402
3242754"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfZGFzaCxMXzIyM18zMl83M19waHlzYg==",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517410139.071,
            "SequenceNumber": "4958128701116962872829005272966843950256545798765
4123522"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfZGFzaCxMXzIyM18zMl83M19iaW9i",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517410146.639,
            "SequenceNumber": "4958128701116962872829005272969382694477736581879
8243842"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfdHNOY25OcixMXzIyM18zMl83M19jaGVtYg==",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517410166.635,
            "SequenceNumber": "4958128701116962872829005272979174993616615215633
8978818"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfdHNOY25OcixMXzIyM18zMl83M19hZ3Jjc2NpYg==",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517411250.903,
            "SequenceNumber": "4958128701116962872829005273503969691911333189565
1270658"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfdHNOY25OcixMXzIyM18zMl83M19hZ3Jjc2NpYQ==",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517411546.631,
            "SequenceNumber": "4958128701116962872829005273646743831207822929194
4378370"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfdHNOY25OcixMXzIyM18zMl83M19hZ3Jjc2E=",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517411555.864,
            "SequenceNumber": "4958128701116962872829005273651700427068242963786
3145474"
        },
        {
            "Data": "TF8xMjNfMzRfNDNfdHNOY25OcixMXzIyM18zMl83M19hZ3JoanloZw==",
            "PartitionKey": "123",
            "ApproximateArrivalTimestamp": 1517411710.075,
            "SequenceNumber": "4958128701116962872829005273726774720466312500687
4574850"
        }
    ],
    "NextShardIterator": "AAAAAAAAAAElquXUpIdyYtD5L8iuWTcmlT+bLzOBQ5gxV1VFEKjbWg
3ObadS74rDxwQYkfA9qnMI9/b78I+7t44rEkZQiPWasTLY+Wi39nsS16YpLYiWPyZbxByAgVS9V5Yp+U
XeO+YXOOQGGt1+jTlWUCYV5BqCwPSzYOOWaqTzoJaTrJO2CuArTknPbiPPaho4TWVIOoQgE7Lph12AqS
6ixYfxUMOp",
    "MillisBehindLatest": 0
}

Elite8300@Elite8300-PC MINGW64 ~/Documents/ServerlessProjects
$
```

Note that if you make a get-records call with a shardID that does not have anything, ***you will have to keep making the calls using the NextShardIterator value until you get to the shard with the available records*** as above



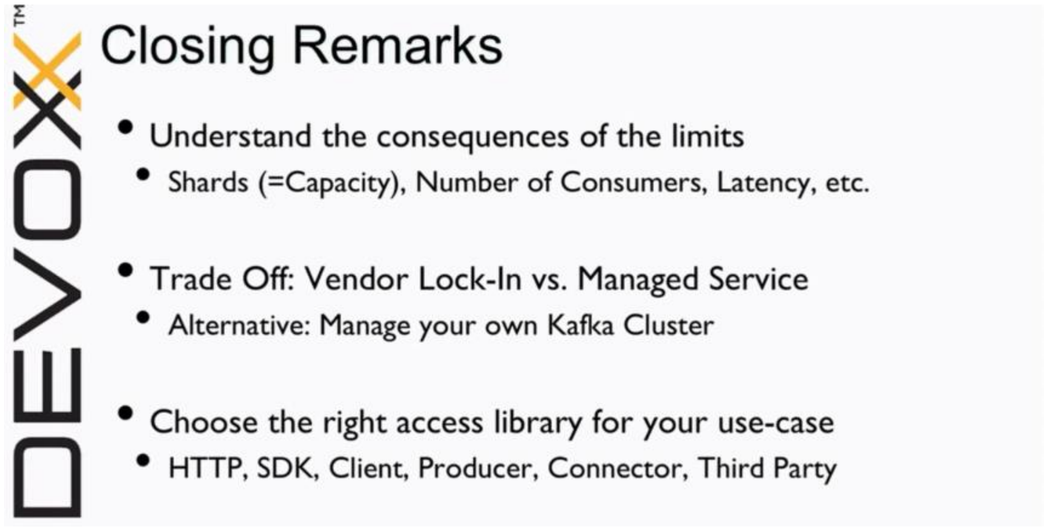
```
Demo $ cat > iterator.tmp
AAAAAAAAAAEqUk9Ckwv03P6xqfZyS0hXCN0TKfCLl05G/jBqKVqgvQOwcxHEEFM+4zSOQW67CWk2uIf+/aErVsmgKcBG6hMXPNCR2txRJR/Xq+Hgqqq3NgLVBORCQbE
vfv3Y6U5B9jEfommncqNer72xsAFfpoTK5HmxMI01rIUW0ioWdatc/yuxph3to4Glldugb3exssFHZrWhjJsUmciYOZ6S7mi
Demo $ step7_get
+ aws kinesis get-records --shard-iterator AAAAAAAAAAEqUk9Ckwv03P6xqfZyS0hXCN0TKfCLl05G/jBqKVqgvQOwcxHEEFM+4zSOQW67CWk2uIf+/aEr
VsmgKcBG6hMXPNCR2txRJR/Xq+Hgqqq3NgLVBORCQbEvfv3Y6U5B9jEfommncqNer72xsAFfpoTK5HmxMI01rIUW0ioWdatc/yuxph3to4Glldugb3exssFHZrWhjJ
sUmciYOZ6S7mi
{
    "Records": [],
    "NextShardIterator": "AAAAAAAAAAEihsOk3xDqpwLF1NtV0Wq/khh7f13bUCJIQaoY5GbPVY895aQV2TZck7/KsjRCcQIU+AK3II3oTYXY9lE+Rgw113r7/
m548PhJ4mBtIPwJpc19PukiGdd5yfxBpAty0oJGGVn1ra+8lUvOwzC8r+f7o/nlIrc2Q2itSqenSFDAEI66Lx7LPTAaG018zcaqZns28M1ENmUdMJbteqz3YFM8",
    "MillisBehindLatest": 0
}
Demo $
```

We can put the Next Iterator into our tmp file again using the command ***cat > iterator.tmp jhgfkugkiutgoytgoytoh***. We can start using it again to get data. Note that the response now is an ***empty Records array*** because there is no record in that shard.



```
Demo $ step9_delete
+ aws kinesis delete-stream --stream-name Devoxx
Demo $ step3_describe
+ aws kinesis describe-stream --stream-name Devoxx
{
    "StreamDescription": {
        "StreamStatus": "DELETING",
        "StreamName": "Devoxx",
        "StreamARN": "arn:aws:kinesis:eu-central-1:127961942231:stream/Devoxx",
        "Shards": []
    }
}
Demo $
```

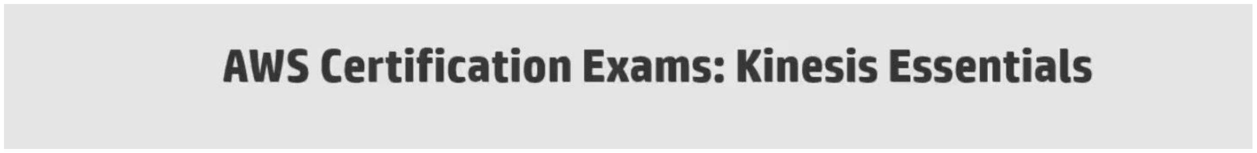
We can then DELETE the stream using the command ***$ aws kinesis delete-stream - -stream-name Devoxx***. We can now see the status is DELETING using the command ***$ aws describe-stream - -stream-name Devoxx***



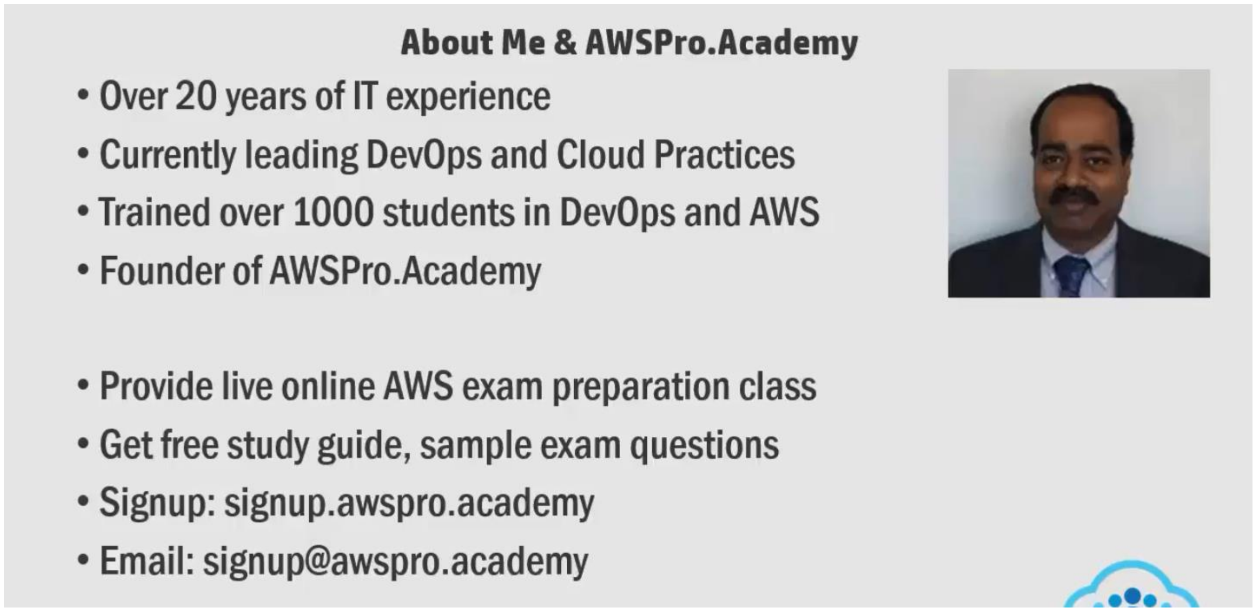
Closing Remarks

- Understand the consequences of the limits
  - Shards (=Capacity), Number of Consumers, Latency, etc.
- Trade Off: Vendor Lock-In vs. Managed Service
  - Alternative: Manage your own Kafka Cluster
- Choose the right access library for your use-case
  - HTTP, SDK, Client, Producer, Connector, Third Party

The Shard is a unit of Capacity and it means there is need for balancing the values used. We have used the CLI in this demo but you will need to use the correct access library for your client.



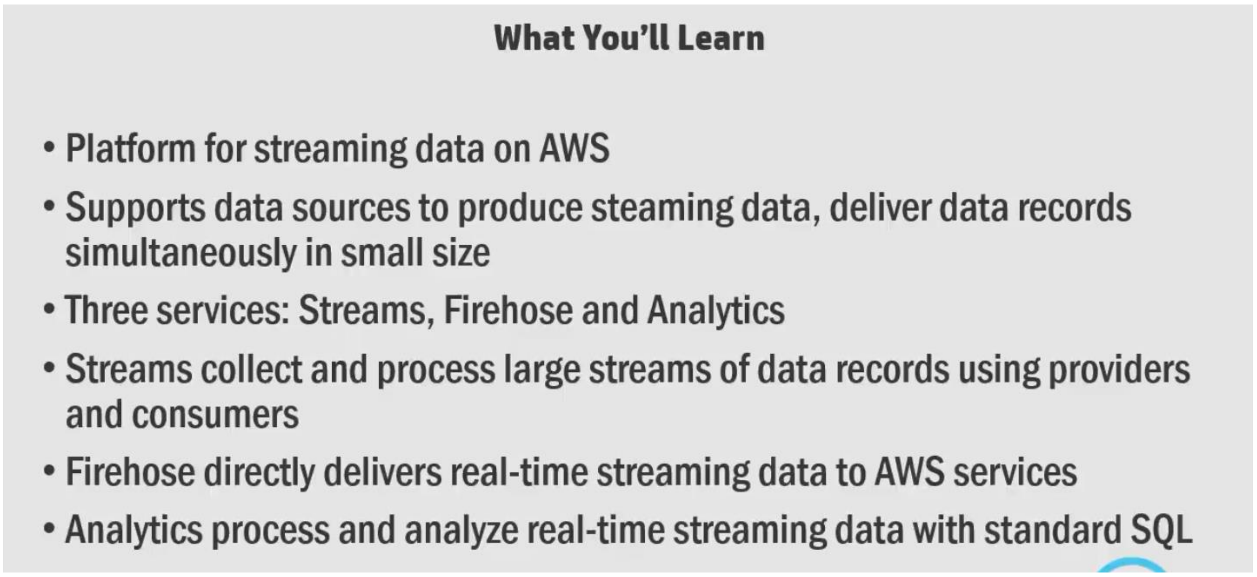
AWS Certification Exams: Kinesis Essentials



About Me & AWSPro.Academy

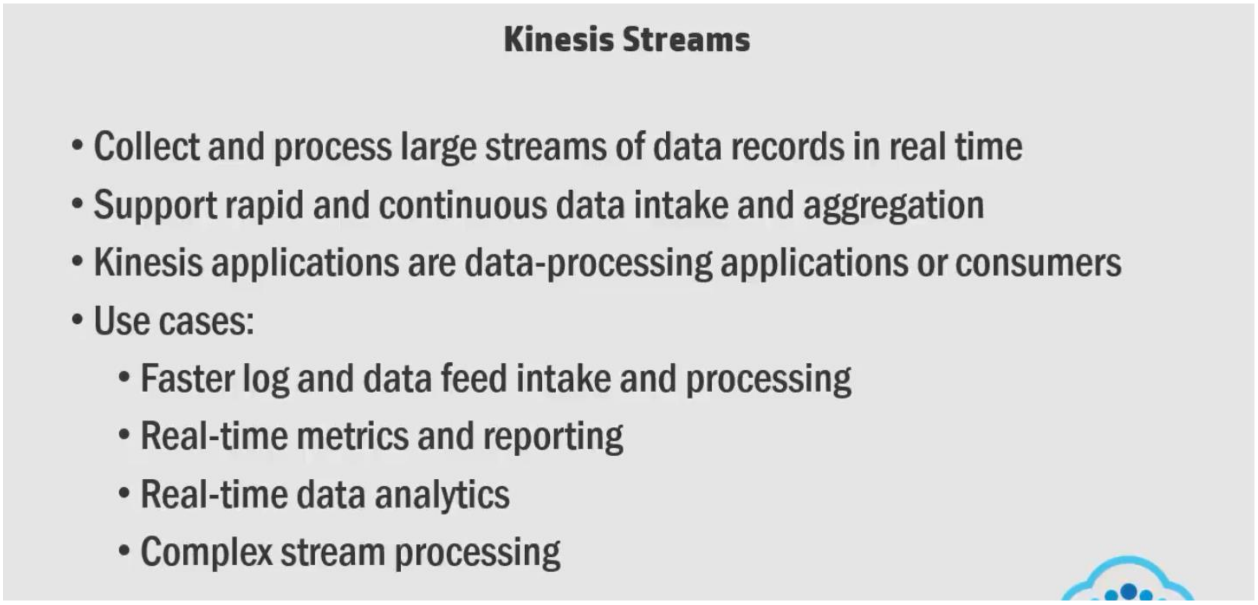
- Over 20 years of IT experience
- Currently leading DevOps and Cloud Practices
- Trained over 1000 students in DevOps and AWS
- Founder of AWSPro.Academy

- Provide live online AWS exam preparation class
- Get free study guide, sample exam questions
- Signup: signup.awspro.academy
- Email: signup@awspro.academy



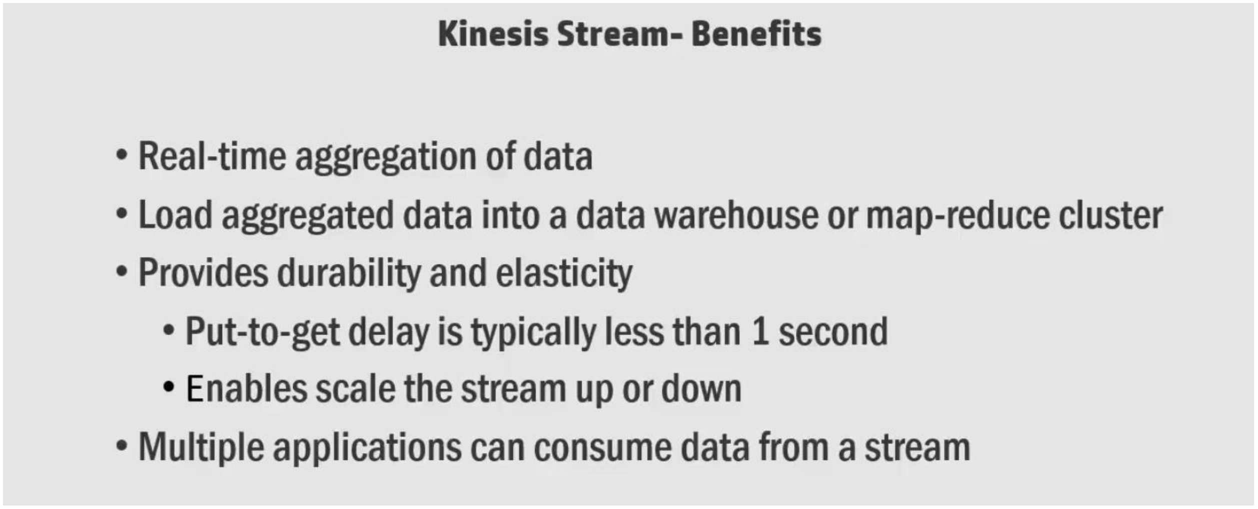
What You'll Learn

- Platform for streaming data on AWS
- Supports data sources to produce steaming data, deliver data records simultaneously in small size
- Three services: Streams, Firehose and Analytics
- Streams collect and process large streams of data records using providers and consumers
- Firehose directly delivers real-time streaming data to AWS services
- Analytics process and analyze real-time streaming data with standard SQL



Kinesis Streams

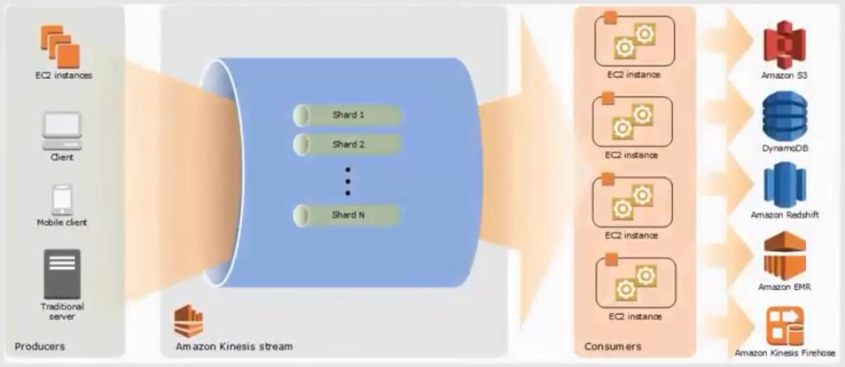
- Collect and process large streams of data records in real time
- Support rapid and continuous data intake and aggregation
- Kinesis applications are data-processing applications or consumers
- Use cases:
  - Faster log and data feed intake and processing
  - Real-time metrics and reporting
  - Real-time data analytics
  - Complex stream processing



Kinesis Stream- Benefits

- Real-time aggregation of data
- Load aggregated data into a data warehouse or map-reduce cluster
- Provides durability and elasticity
  - Put-to-get delay is typically less than 1 second
  - Enables scale the stream up or down
- Multiple applications can consume data from a stream

Kinesis Streams - Architecture



Kinesis stream - Shards

- Streams are made of shards, and used as base throughput unit of a stream
- Write: Each shard supports up to 1,000 records/sec or up to maximum rate of 1 MB/sec
- Read: Each shard support up to 5 transactions/sec or up to a maximum read rate of 2 MB/sec
- PUT data call will be rejected with ProvisionedThroughputExceeded exception When throughput limits are exceeded
- Record's retention period is set to a default of 24 hours after creation, but can be extended up to 7 days

Kinesis stream - Records

- Stream is an ordered sequence of data records. A record is the unit of data stored in a stream
- Record is composed of a sequence number, partition key, and data blob
- Data blob is the original data from a producer with maximum size of 1 MB
- Partition key helps to identify and route records to different shards
- A sequence number is a unique identifier for each record
- Data records are added using API calls (PutRecord and PutRecords) or Amazon Kinesis Producer Library (KPL), or Amazon Kinesis Agent