

Pull and Consume Data from Stream

To Begin with the Lab

Summary of the Lab

This lab demonstrates how to pull and consume data from an AWS Kinesis Data Stream using the AWS CLI. You manually send records to the stream, retrieve the ShardIterator, and then use it to fetch records. The output displays shard details and encoded data, which can be decoded to view the original record contents.

- **Prerequisite**
 - An **active Kinesis Data Stream** (e.g., my-first-data-stream) created earlier.
 - The **AWS CLI** installed and configured (aws configure).
 - Necessary IAM permissions for Kinesis operations (AmazonKinesisFullAccess).
- We will manually send records into your stream using the **AWS CLI**.



- The output will include:
- **ShardId**: the shard receiving the data
- **SequenceNumber**: unique identifier within the shard

```
CloudShell
us-east-1
$ aws kinesis put-record --stream-name myfirstdatastream --partition-key "PartitionKey" --data $(echo -n "Data Entry 1" | base64)
{
  "ShardId": "shardId-000000000001",
  "SequenceNumber": "49668282451566395191788156861457786746359645183751954458"
}
```

```
CloudShell
us-east-1
$ aws kinesis put-record --stream-name myfirstdatastream --partition-key "PartitionKey" --data $(echo -n "Data Entry 1" | base64)
{
  "ShardId": "shardId-000000000001",
  "SequenceNumber": "49668282451566395191788156861457786746359645183751954458"
}
$ aws kinesis put-record --stream-name myfirstdatastream --partition-key "PartitionKey2" --data $(echo -n "Data Entry 2" | base64)
{
  "ShardId": "shardId-000000000001",
  "SequenceNumber": "49668282451566395191788156861458995672179268883897589778"
}
$ aws kinesis put-record --stream-name myfirstdatastream --partition-key "PartitionKey3" --data $(echo -n "Data Entry 3" | base64)
{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49668282451544894446589626238318668879726235896288636354"
}
```

- Use this command to get the ShardIterator and make sure to change your shard-id
- `aws kinesis get-shard-iterator --stream-name myfirstdatastream --shard-id "shardId-000000000001" --shard-iterator-type LATEST`

```
$ aws kinesis get-shard-iterator --stream-name myfirstdatastream --shard-id "shardId-000000000001" --shard-iterator-type LATEST
{
  "ShardIterator": "PAAAAAAAAAAGsg4TAqcwDLcxd5WIAv7Zbucrw7E14zEz7n3ac2TDwe23WQ1LPobM8Nz2ym8myND+9RxyALTC6zxyBfK4ZD28ZLNLznToInka36/1LrBAlNd8Ipcrc1SS8S33h+qfyjnfT3Uht39m860+b10HGoKvxdvduRPP6A76AFu8mzLAFgetmbat/exL4Zq:Yu31G6TDv8KfQ3uFn37Ru21Nc3sL7g91Am!"
}
```

- And to get your record use the command
Change your "YourShardIterator" to the real ShardIterator that you get from the previous command
- `aws kinesis get-records --shard-iterator "ShardIterator"`
- After running the command, you will get the output with the records.

```
CloudShell
us-east-1

$ aws kinesis put-record --stream-name myfirstdatastream --partition-key "PartitionKey" --data $(echo -n "Data Entry 1" | base64)
{"ShardId": "shardid-000000000001", "SequenceNumber": "49668282451566395191788156861457786746359645103751954459"}
$ aws kinesis put-record --stream-name myfirstdatastream --partition-key "PartitionKey2" --data $(echo -n "Data Entry 2" | base64)
{"ShardId": "shardid-000000000001", "SequenceNumber": "49668282451566395191788156861458995672179268883897589778"}
$ aws kinesis put-record --stream-name myfirstdatastream --partition-key "PartitionKey3" --data $(echo -n "Data Entry 3" | base64)
{"ShardId": "shardid-000000000000", "SequenceNumber": "49668282451566395191788156861458995672179268883897589778"}
$ aws kinesis get-shard-iterator --stream-name myfirstdatastream --shard-id "shardid-000000000001" --shard-iterator-type LATEST
{"ShardIterator": "AAAAAAAAAFKw6P8R4rDU7Eb1ogQcETZb1c2uNDu/N/GSCbcPHY9csTWC3uJ08B3Z9uXezNMGxx9xfzh1unkE7C2DMLT0QpZGrq819rb9/V8DfQyncR6Qh7A3CQLz4hn1mwJh9q1+Lz0YtcowLLnugJCw2XpAmlJJAqr14N6nP25n/BM41UBNB5q7UD93K2x2ap87c18t1qU5kf310Lc0G6Anp1J1CT7P6Ga1ABM6457Qwe=="}
$ aws kinesis get-records --shard-iterator "AAAAAAAAAFKw6P8R4rDU7Eb1ogQcETZb1c2uNDu/N/GSCbcPHY9csTWC3uJ08B3Z9uXezNMGxx9xfzh1unkE7C2DMLT0QpZGrq819rb9/V8DfQyncR6Qh7A3CQLz4hn1mwJh9q1+Lz0YtcowLLnugJCw2XpAmlJJAqr14N6nP25n/BM41UBNB5q7UD93K2x2ap87c18t1qU5kf310Lc0G6Anp1J1CT7P6Ga1ABM6457Qwe=="
{"Records": [{"SequenceNumber": "49668282451566395191788156861457786746359645103751954459", "ApproximateArrivalTimestamp": "2025-10-23T12:38:15.939800+00:00", "Data": "RGF0YSBFbnRyeSAx", "PartitionKey": "PartitionKey"}], "NextShardIterator": "AAAAAAAAAGBPIAUyfy3L72YalB9VsXvM3Rsj1Y1l0FvXmCzvlBacWuHSPAlTf0VU783PVCvkEJnvmL1vBektZTqdGhAjjcraCAHAv9z3KttldVx18KJFSTLupFXTsXECQL411R36FVoxQfC5Qh66MCBES2kAUfBKR5US1nfGu1h194zg7WuycF488RVd5fmb0fr39Po1QvzGrM80+VpQhAUCBrCzbpC8SeQX04ezgyPI1kt1Gkz6L7tPKIOg9AL5rKecAUjSkPlalFdqldQVYSATz0+LoH2Z1ZXVMP0t==", "MillisBehindLatest": 2648000}}
$ aws kinesis put-record --stream-name myfirstdatastream --partition-key "PartitionKey" --data $(echo -n "Data Entry 1b" | base64)
{"ShardId": "shardid-000000000001", "SequenceNumber": "496682828330882223541818078965169847485756433925931026"}
$ aws kinesis get-shard-iterator --stream-name myfirstdatastream --shard-id "shardid-000000000001" --shard-iterator-type LATEST
{"ShardIterator": "AAAAAAAAAGBPIAUyfy3L72YalB9VsXvM3Rsj1Y1l0FvXmCzvlBacWuHSPAlTf0VU783PVCvkEJnvmL1vBektZTqdGhAjjcraCAHAv9z3KttldVx18KJFSTLupFXTsXECQL411R36FVoxQfC5Qh66MCBES2kAUfBKR5US1nfGu1h194zg7WuycF488RVd5fmb0fr39Po1QvzGrM80+VpQhAUCBrCzbpC8SeQX04ezgyPI1kt1Gkz6L7tPKIOg9AL5rKecAUjSkPlalFdqldQVYSATz0+LoH2Z1ZXVMP0t=="}
$
```

- There was an entry that we have decoded

```
us-east-1

$ aws kinesis get-records --shard-iterator "AAAAAAAAAGBPIAUyfy3L72YalB9VsXvM3Rsj1Y1l0FvXmCzvlBacWuHSPAlTf0VU783PVCvkEJnvmL1vBektZTqdGhAjjcraCAHAv9z3KttldVx18KJFSTLupFXTsXECQL411R36FVoxQfC5Qh66MCBES2kAUfBKR5US1nfGu1h194zg7WuycF488RVd5fmb0fr39Po1QvzGrM80+VpQhAUCBrCzbpC8SeQX04ezgyPI1kt1Gkz6L7tPKIOg9AL5rKecAUjSkPlalFdqldQVYSATz0+LoH2Z1ZXVMP0t=="
{"Records": [{"SequenceNumber": "49668282451566395191788156861457786746359645103751954459", "ApproximateArrivalTimestamp": "2025-10-23T12:38:15.939800+00:00", "Data": "RGF0YSBFbnRyeSAx", "PartitionKey": "PartitionKey"}], "NextShardIterator": "AAAAAAAAAGBPIAUyfy3L72YalB9VsXvM3Rsj1Y1l0FvXmCzvlBacWuHSPAlTf0VU783PVCvkEJnvmL1vBektZTqdGhAjjcraCAHAv9z3KttldVx18KJFSTLupFXTsXECQL411R36FVoxQfC5Qh66MCBES2kAUfBKR5US1nfGu1h194zg7WuycF488RVd5fmb0fr39Po1QvzGrM80+VpQhAUCBrCzbpC8SeQX04ezgyPI1kt1Gkz6L7tPKIOg9AL5rKecAUjSkPlalFdqldQVYSATz0+LoH2Z1ZXVMP0t==", "MillisBehindLatest": 2648000}}
$
```

- Copy and paste the Data Value here and click on the Decode.

Decode from Base64 format

Simply enter your data then push the decode button.

RGF0YSBFbnRyeSAx

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8

Source character set.

☐

Decode each line separately (useful for when you have multiple entries).

☒ Live mode OFF

Decodes in real-time as you type or paste (supports only the UTF-8 character set).

< DECODE >

Decodes your data into the area below.

Data Entry 1