

Developing a streaming data ecosystem using AWS Kinesis

- Contents:

- Problem of streaming data

- Kinesis capabilities

- Example project

- Pieces required
 - Setup
 - Examples

- Nabeel Tariq



EMORY

GOIZUETA
BUSINESS
SCHOOL

What makes high velocity data unique

- Analysis is done ad hoc, you don't get to wait for all of the data to perform analysis
- Data often needs transformation before analysis, but both have to be done on the fly
- Data-driven-decisions also made in real time – can't take your time and evaluate multiple analyses



What can you do with AWS

EC2



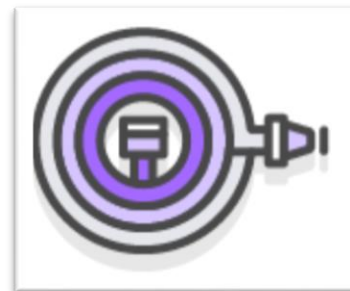
Distributed
search and
analytics engine

Kinesis



Streams

Build custom
applications to
process
streaming data



Firehose

Load streaming
data into AWS

DynamoDB



A fully managed
cloud NoSQL
database

S3



Object storage
built to store
and retrieve
data



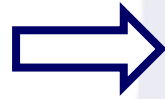
EMORY

GOIZUETA
BUSINESS
SCHOOL

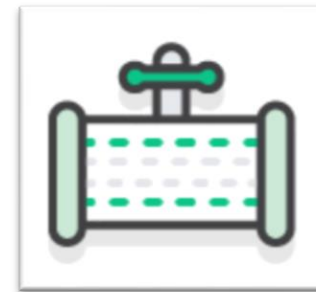
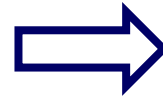
An example of processing streaming data from Twitter



EC2



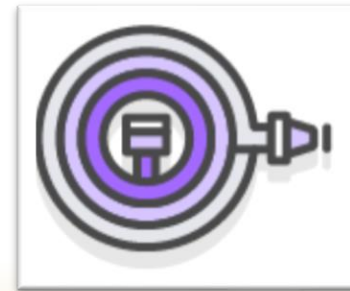
Twitter API



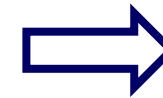
Kinesis
Streams



DynamoDB



Kinesis
Firehose



S3

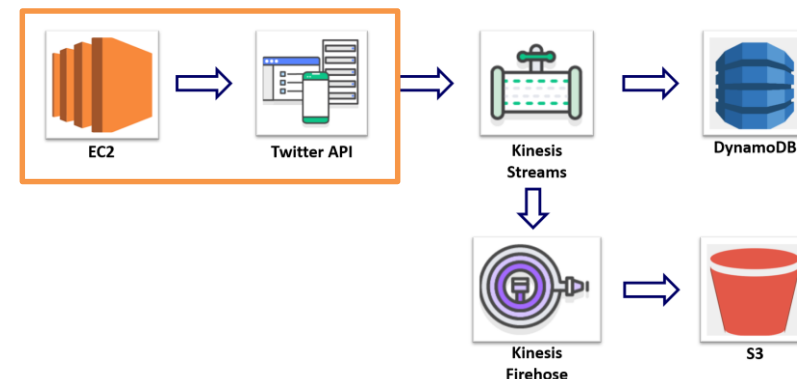


EMORY

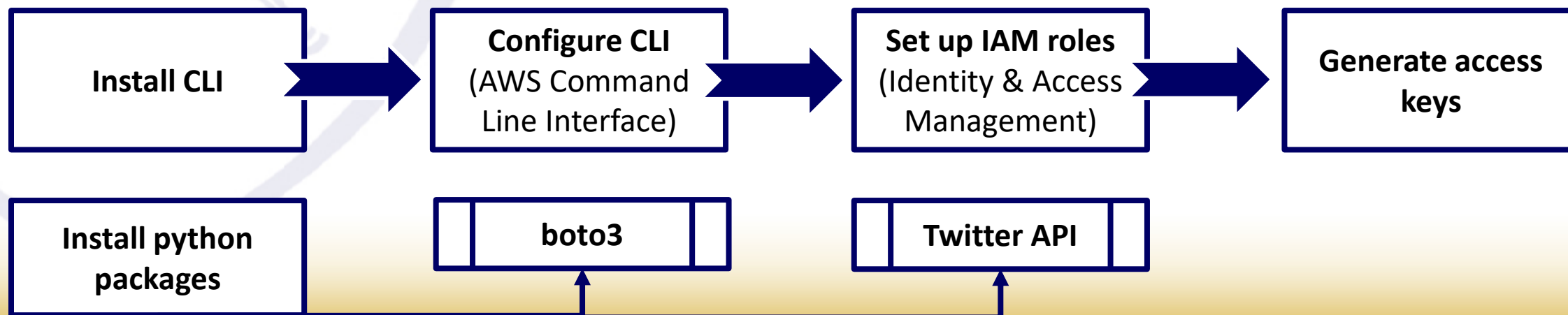
GOIZUETA
BUSINESS
SCHOOL

Setting up properly is essential

Deployed a t2-Micro AWS EC2 instance to host the code and interface with other AWS tools

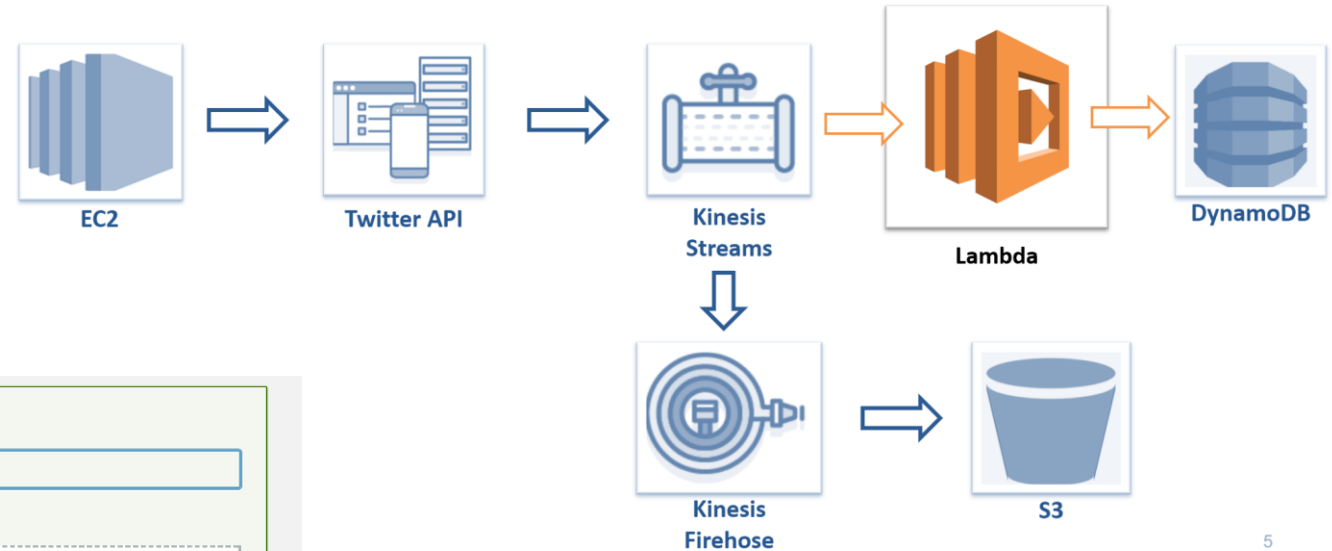


Get software and libraries prepared



Things don't always go according to plan

Our original plan is to build a Lambda function and attach it to the Kinesis Stream for processing the JSON data into a more usable format.



Succeed in running in the test



Get error on live data



Go for DynamoDB

NoSQL structure is more flexible for data processing

✓ Execution result: succeeded ([logs](#))

▼ Details

The area below shows the result returned by your function execution.

22

Summary

Code SHA-256
F1yn3+gGpgASaw674
a2PMc2byRDXyqZ8KG
PDzAb9uBQ=

Request ID
e9e15d29-c9df-11e7-
8173-7f4080138e32

Log output

The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
START RequestId: e9e15d29-c9df-11e7-8173-7f4080138e32 Version: $LATEST
END RequestId: e9e15d29-c9df-11e7-8173-7f4080138e32
REPORT RequestId: e9e15d29-c9df-11e7-8173-7f4080138e32  Duration: 0.39 ms    Billed
Duration: 100 ms    Memory Size: 128 MB    Max Memory Used: 22 MB
```

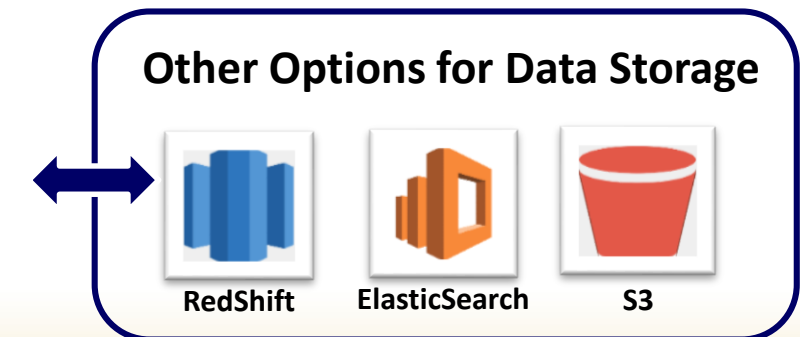
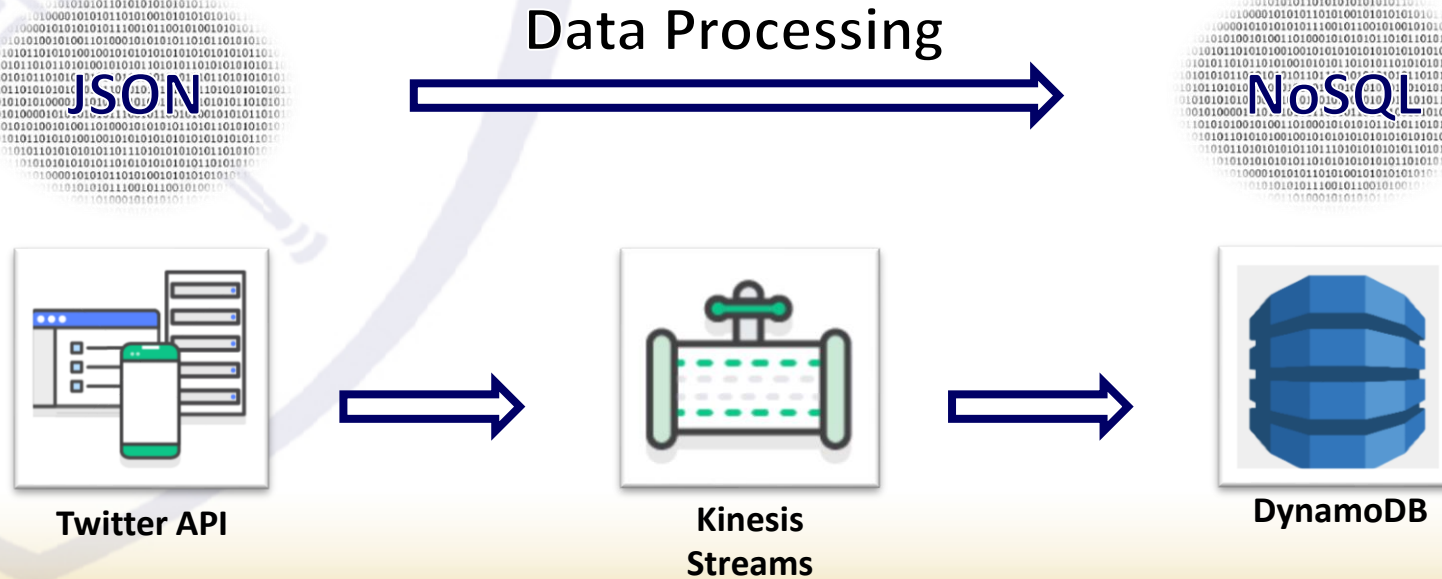
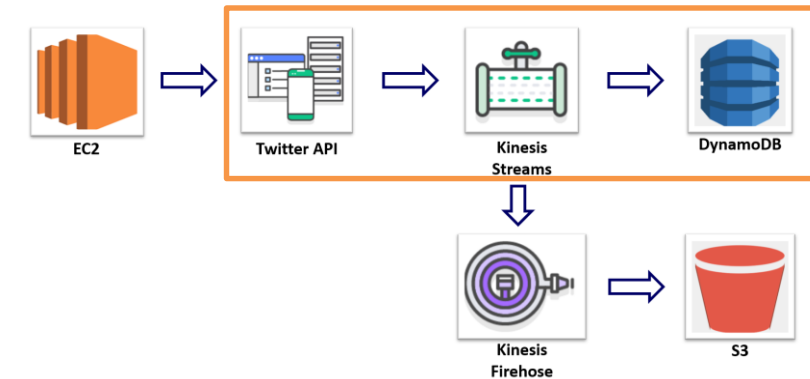


EMORY

GOIZUETA
BUSINESS
SCHOOL

Eventually this is what the process looked like

Connect the Twitter API to AWS using Kinesis Streams, process the streams to extract our attributes, and have the stream feed into DynamoDB





Questions?

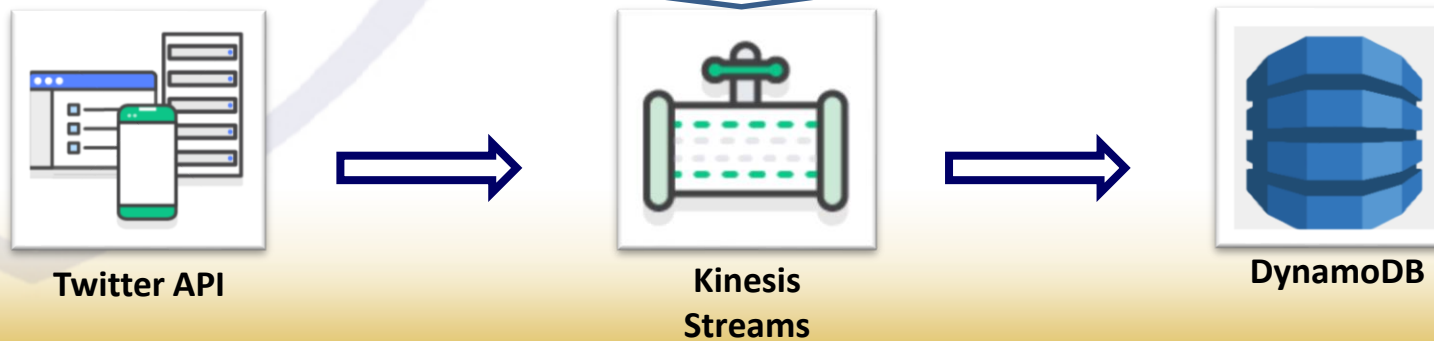
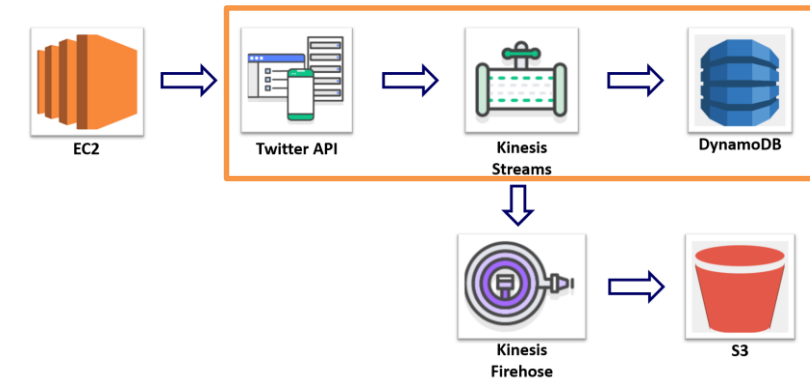
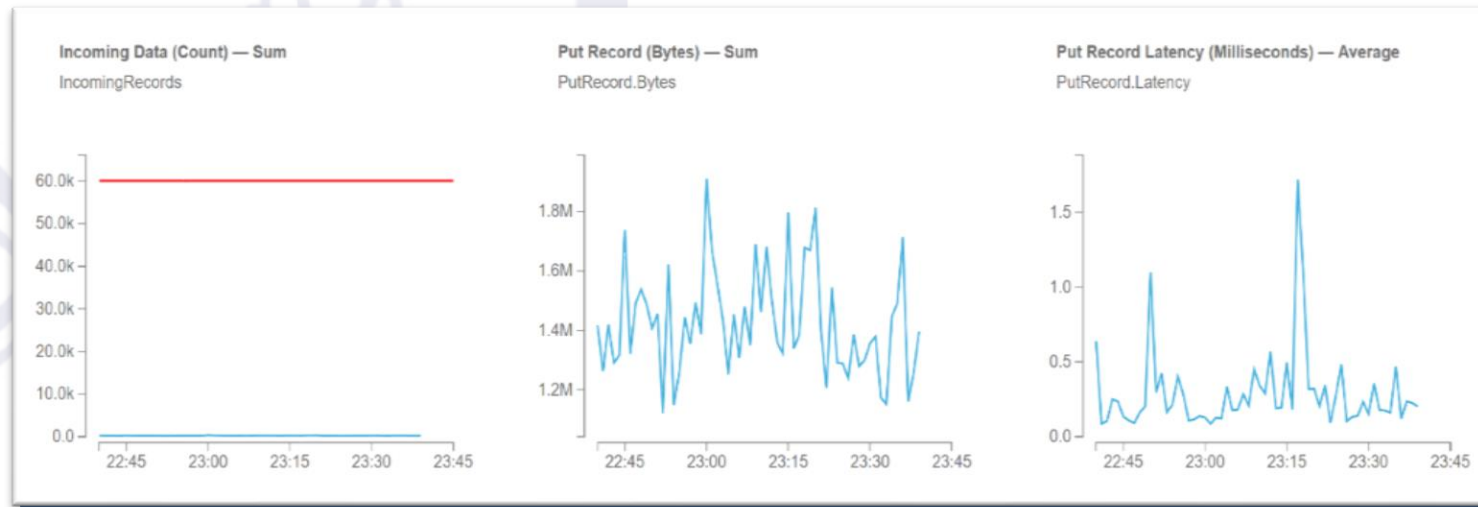


EMORY

GOIZUETA
BUSINESS
SCHOOL

Step 2

AWS Cloud Watch Metric

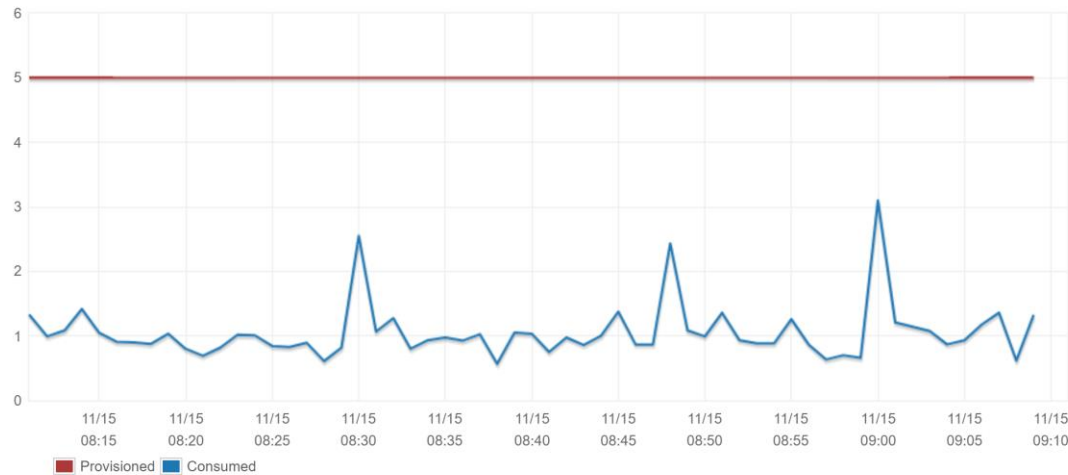


Monitoring

CloudWatch Monitoring Details

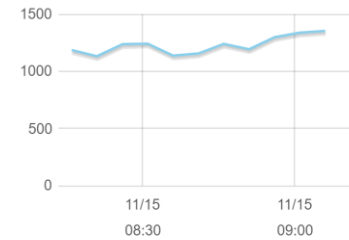
Read capacity (Count)

Statistic: Time Range: Last Hour Period: 1 Minute

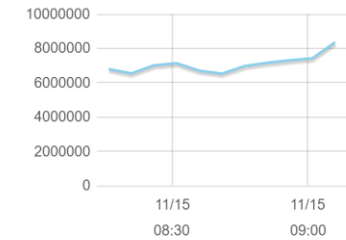


Firehose Console Metrics for Kinesis Stream read

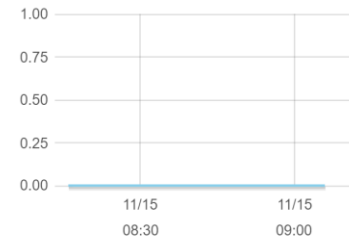
DataReadFromKinesisStream Records (Sum)



DataReadFromKinesisStream Bytes (Sum)

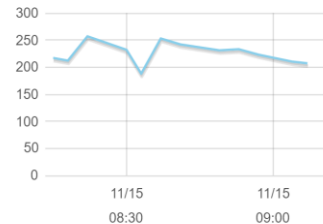


KinesisMillisBehindLatest (Maximum)

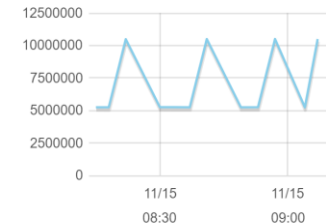


Firehose Console Metrics for S3 write

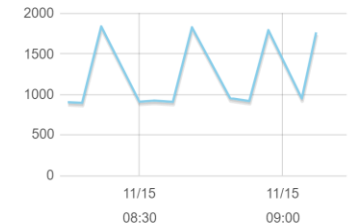
DeliveryToS3 DataFreshness (Maximum)



DeliveryToS3 Bytes (Sum)



DeliveryToS3 Records (Sum)



EMORY

GOIZUETA
BUSINESS
SCHOOL