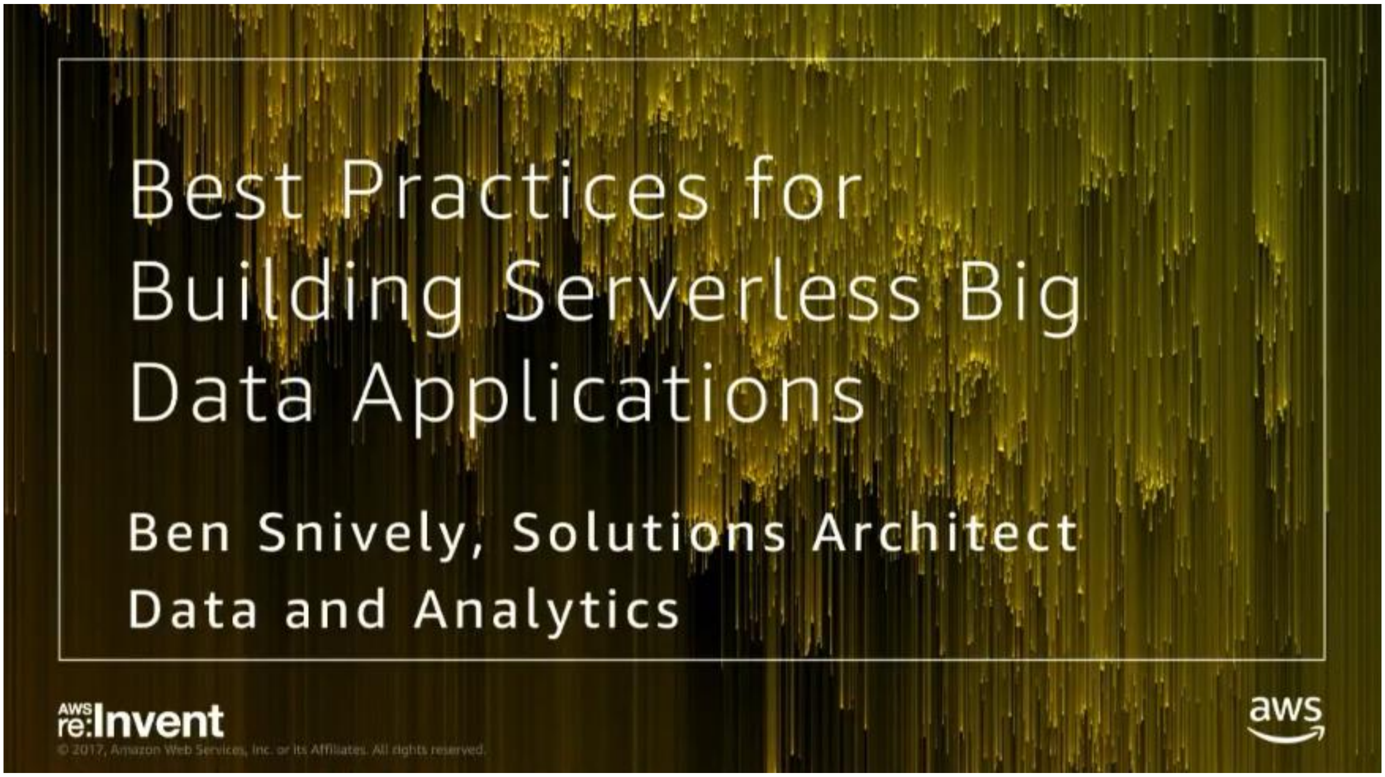


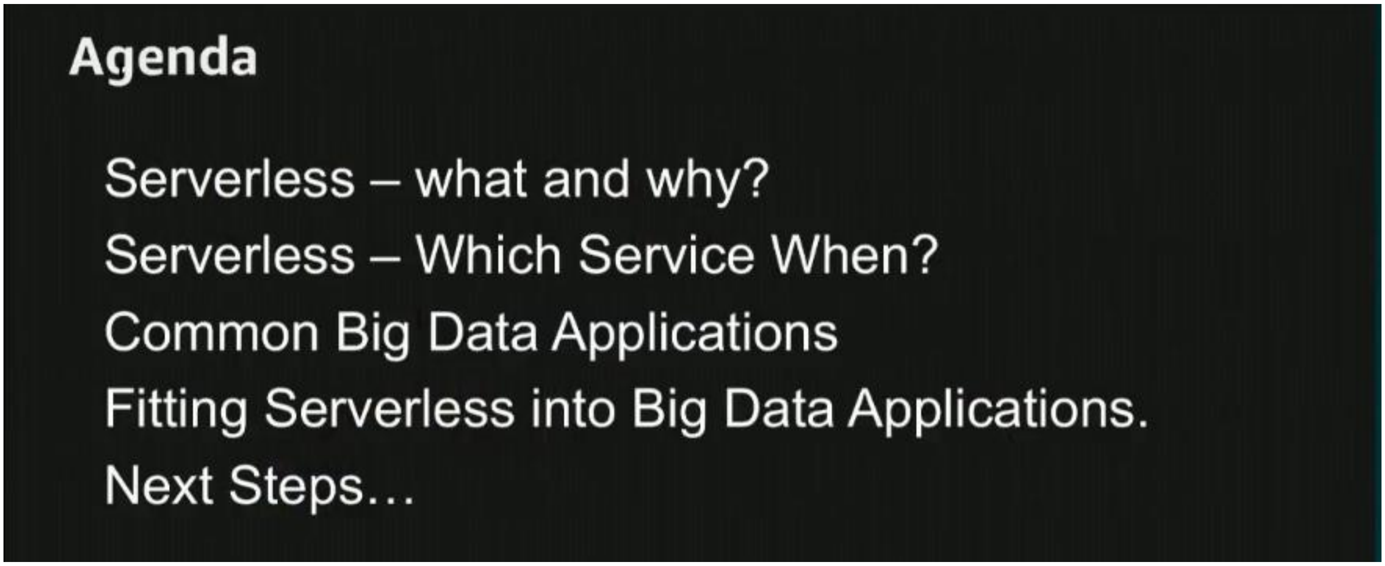
Best Practices for Building Serverless Big Data Applications

Ben Snively, Solutions Architect
Data and Analytics

AWS re:Invent

aws

Serverless technologies let you build and scale applications and services rapidly without the need to provision or manage servers. In this session, we show you how to incorporate serverless concepts into your big data architectures. We explore the concepts behind and benefits of serverless architectures for big data, looking at design patterns to ingest, store, process, and visualize your data. Along the way, we explain when and how you can use serverless technologies to streamline data processing, minimize infrastructure management, and improve agility and robustness and share a reference architecture using a combination of cloud and open source technologies to solve your big data problems. Topics include: use cases and best practices for serverless big data applications; leveraging AWS technologies such as Amazon DynamoDB, Amazon S3, Amazon Kinesis, AWS Lambda, Amazon Athena, and Amazon EMR; and serverless ETL, event processing, ad hoc analysis, and real-time analytics.



Agenda

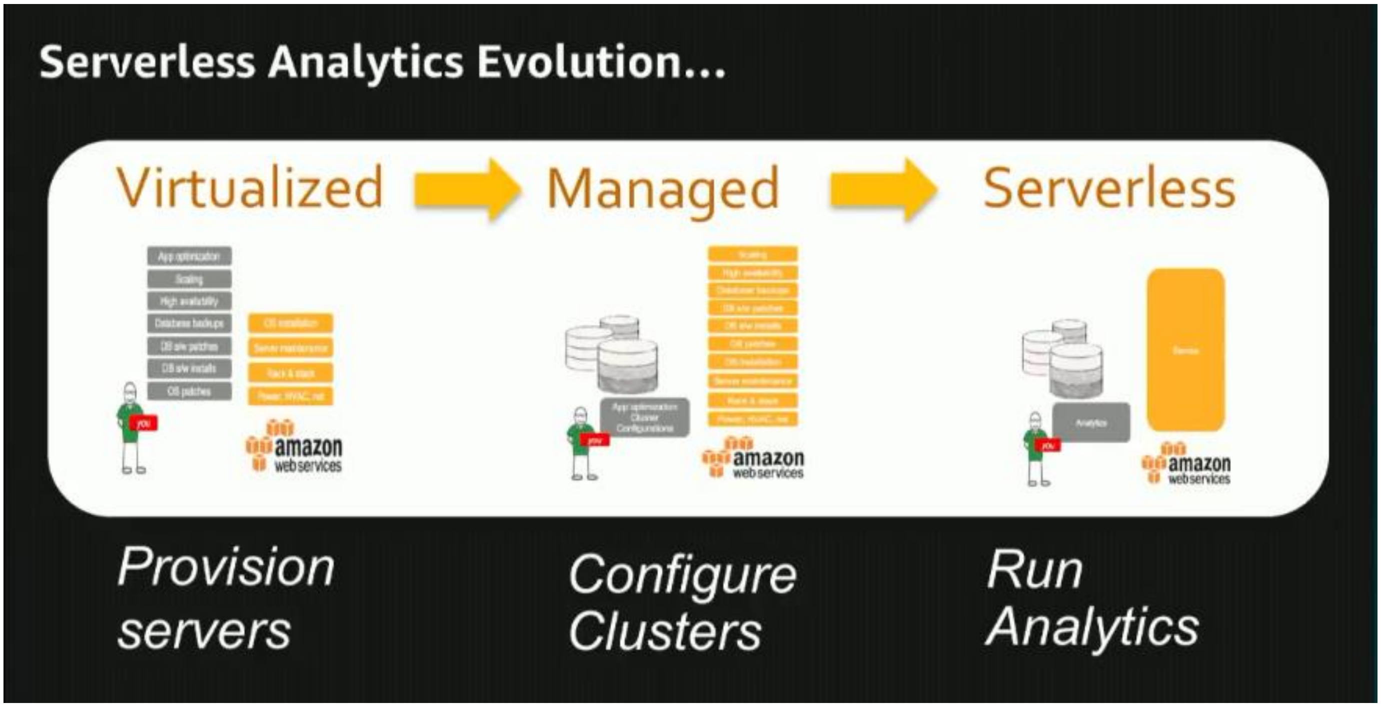
Serverless – what and why?

Serverless – Which Service When?

Common Big Data Applications

Fitting Serverless into Big Data Applications.

Next Steps…



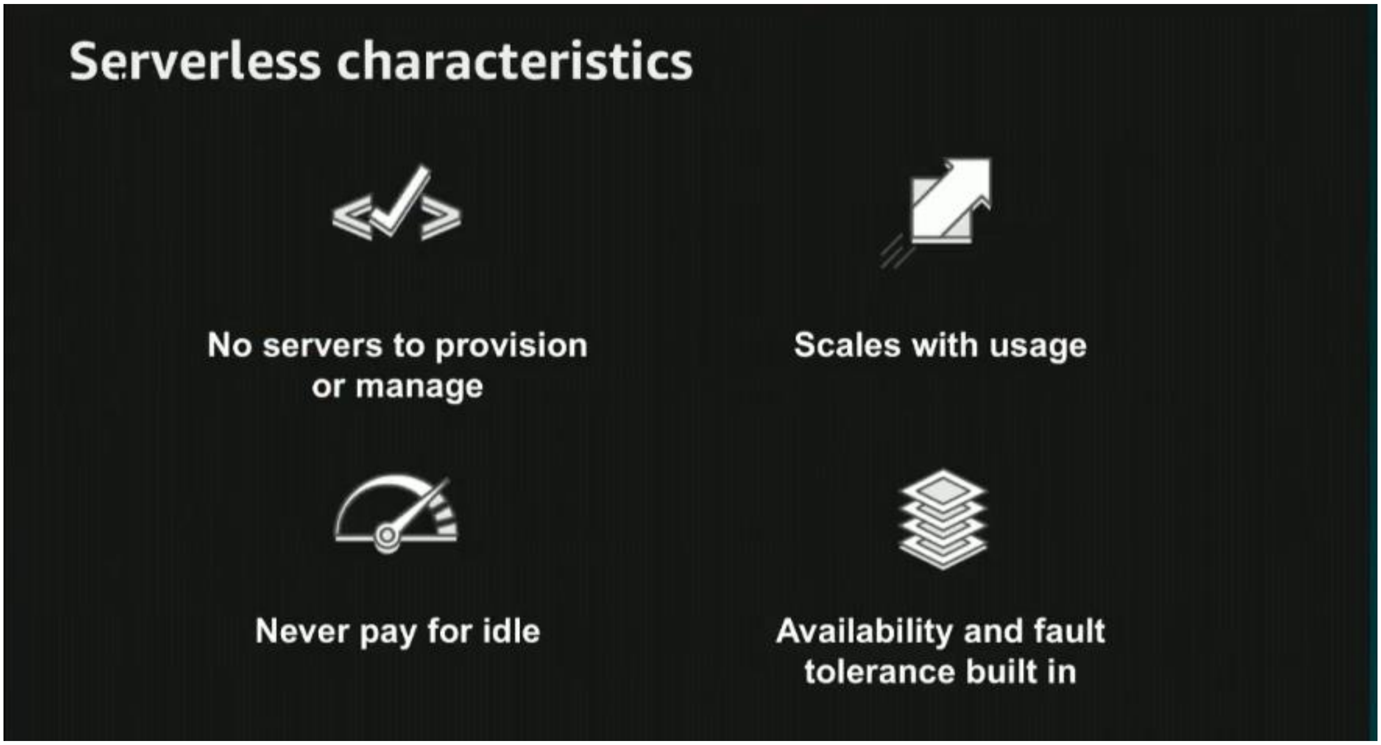
Serverless Analytics Evolution…

Virtualized → Managed → Serverless

Provision servers

Configure Clusters

Run Analytics



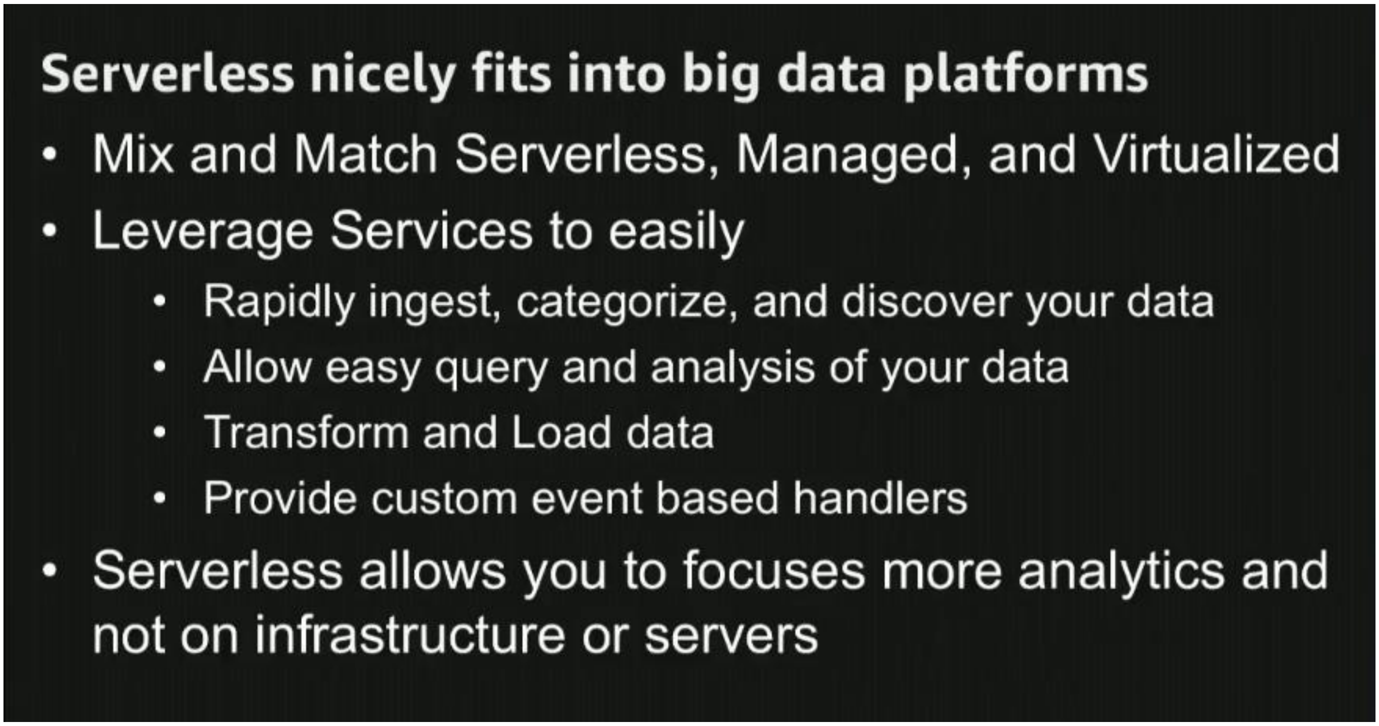
Serverless characteristics

No servers to provision or manage

Scales with usage

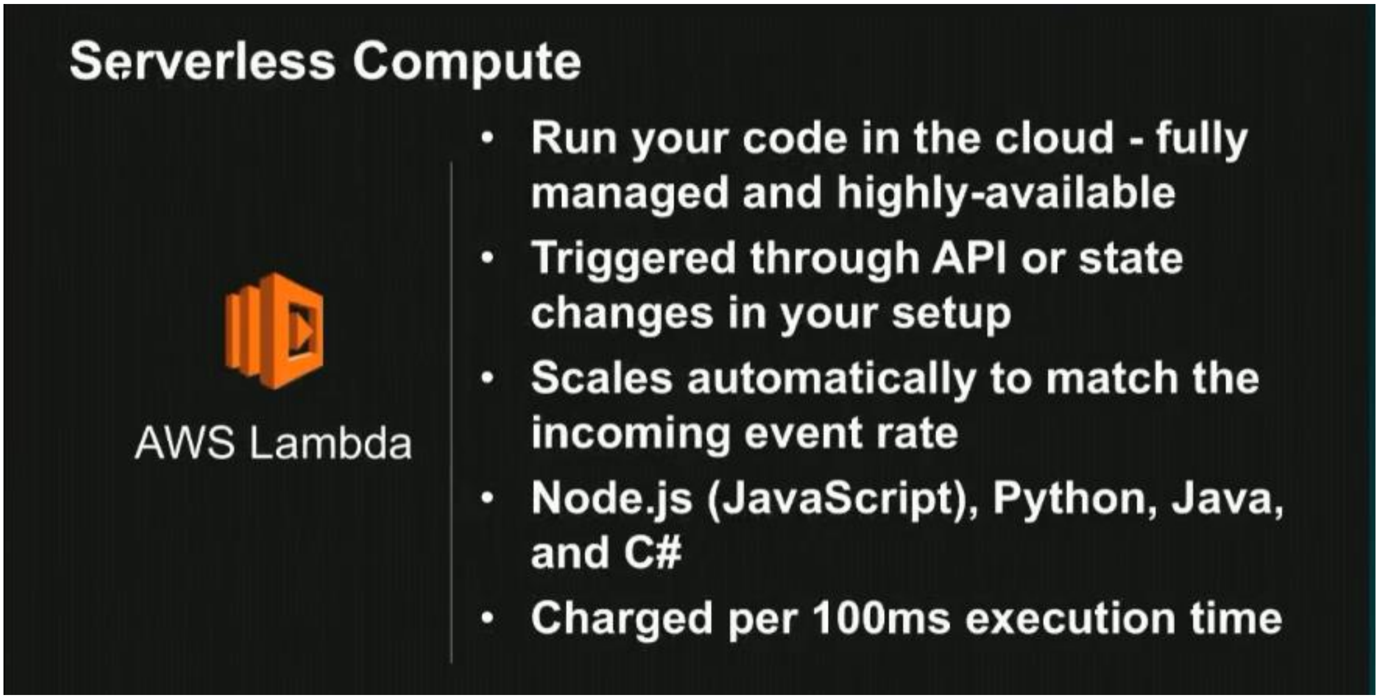
Never pay for idle

Availability and fault tolerance built in



Serverless nicely fits into big data platforms

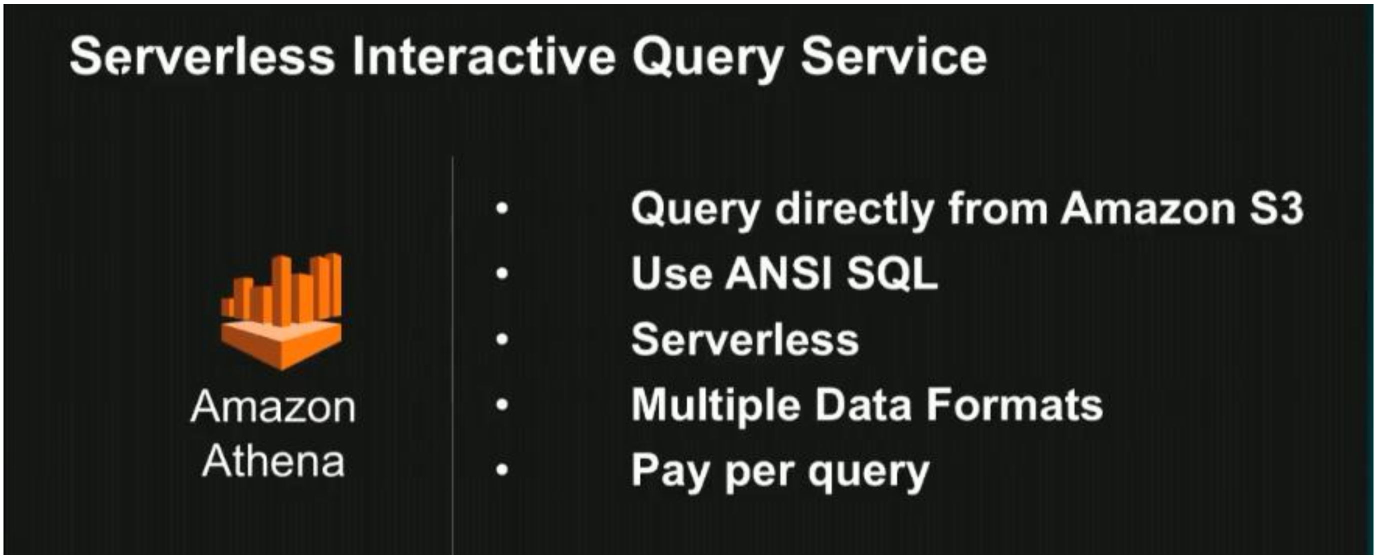
- Mix and Match Serverless, Managed, and Virtualized
- Leverage Services to easily
  - Rapidly ingest, categorize, and discover your data
  - Allow easy query and analysis of your data
  - Transform and Load data
  - Provide custom event based handlers
- Serverless allows you to focuses more analytics and not on infrastructure or servers



Serverless Compute

AWS Lambda

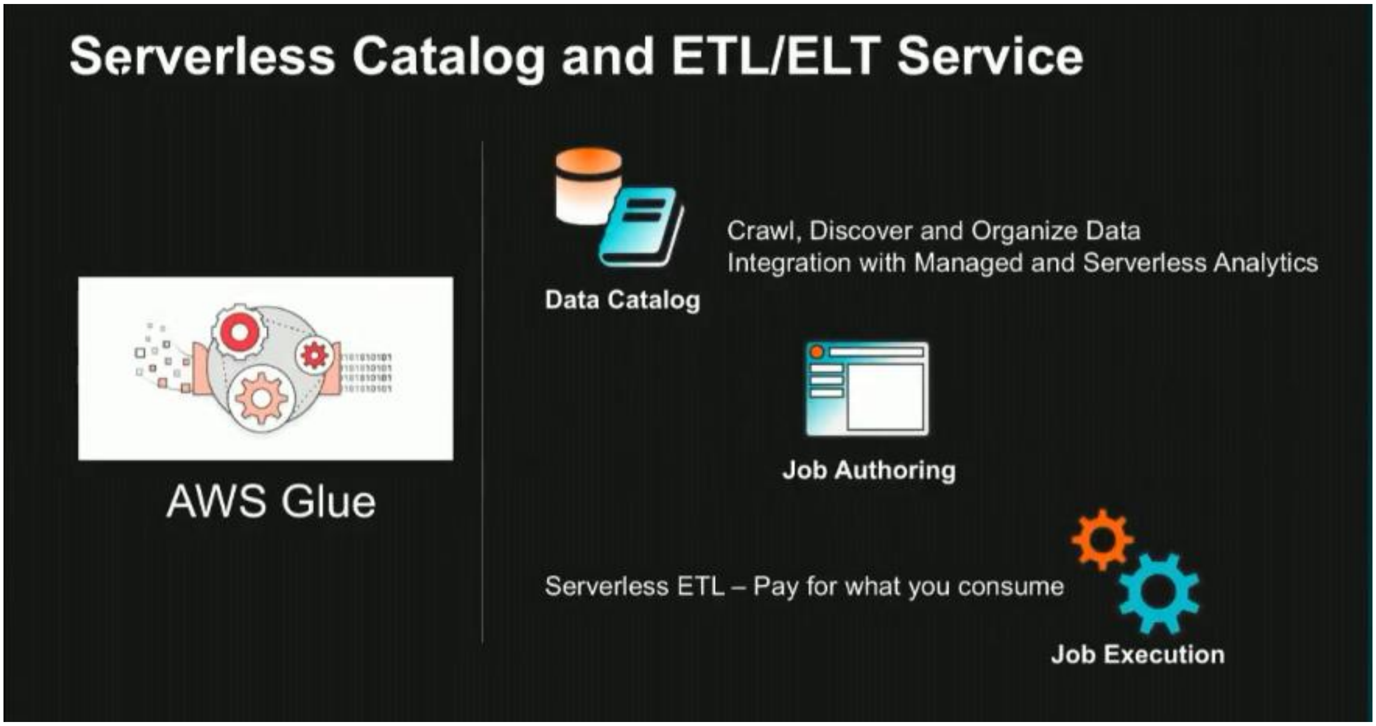
- **Run your code in the cloud - fully managed and highly-available**
- **Triggered through API or state changes in your setup**
- **Scales automatically to match the incoming event rate**
- **Node.js (JavaScript), Python, Java, and C#**
- **Charged per 100ms execution time**



Serverless Interactive Query Service

Amazon Athena

- Query directly from Amazon S3
- Use ANSI SQL
- Serverless
- Multiple Data Formats
- Pay per query



Serverless Catalog and ETL/ELT Service

AWS Glue

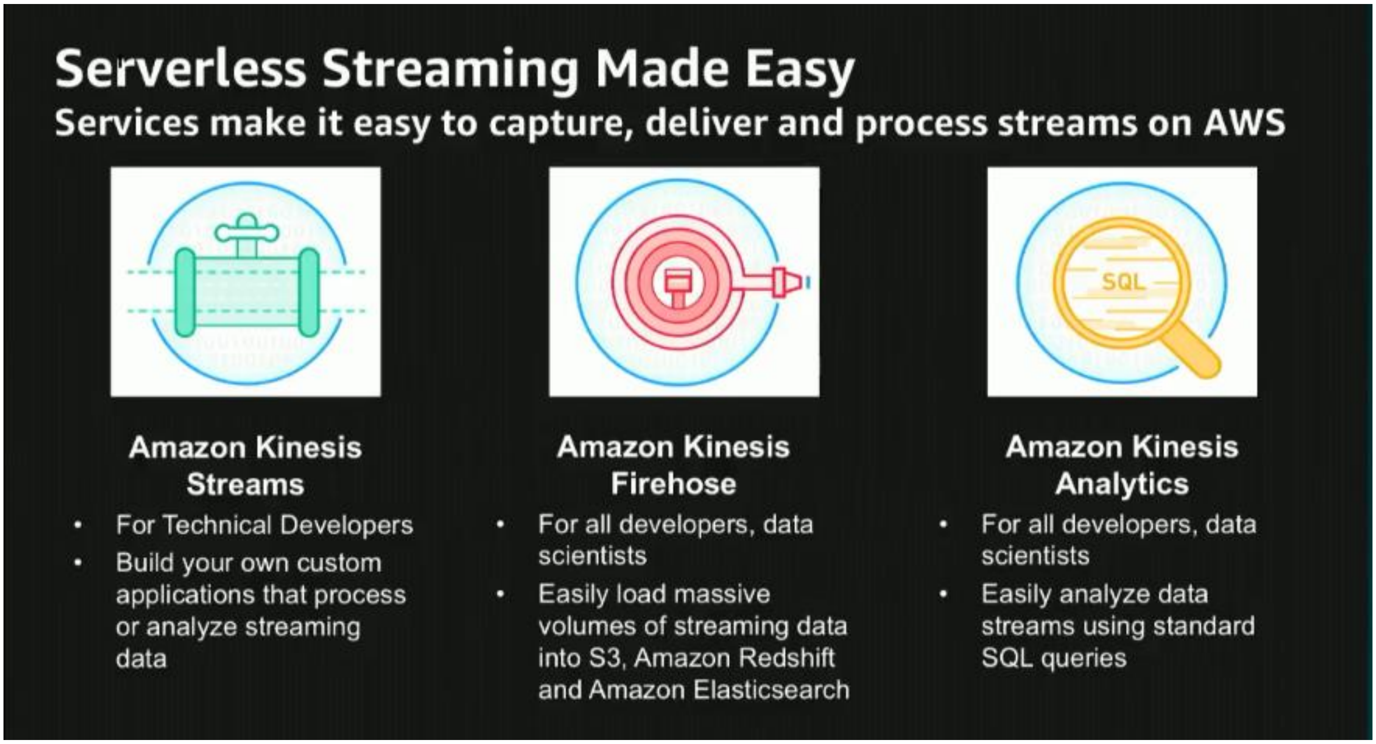
Data Catalog

Crawl, Discover and Organize Data
Integration with Managed and Serverless Analytics

Job Authoring

Serverless ETL – Pay for what you consume

Job Execution



Serverless Streaming Made Easy

Services make it easy to capture, deliver and process streams on AWS

Amazon Kinesis Streams

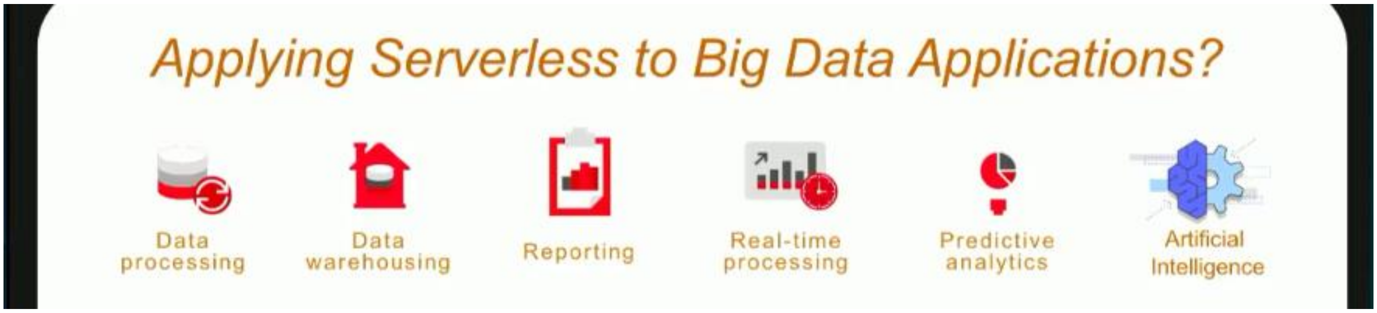
- For Technical Developers
- Build your own custom applications that process or analyze streaming data

Amazon Kinesis Firehose

- For all developers, data scientists
- Easily load massive volumes of streaming data into S3, Amazon Redshift and Amazon Elasticsearch

Amazon Kinesis Analytics

- For all developers, data scientists
- Easily analyze data streams using standard SQL queries



Applying Serverless to Big Data Applications?

Data processing

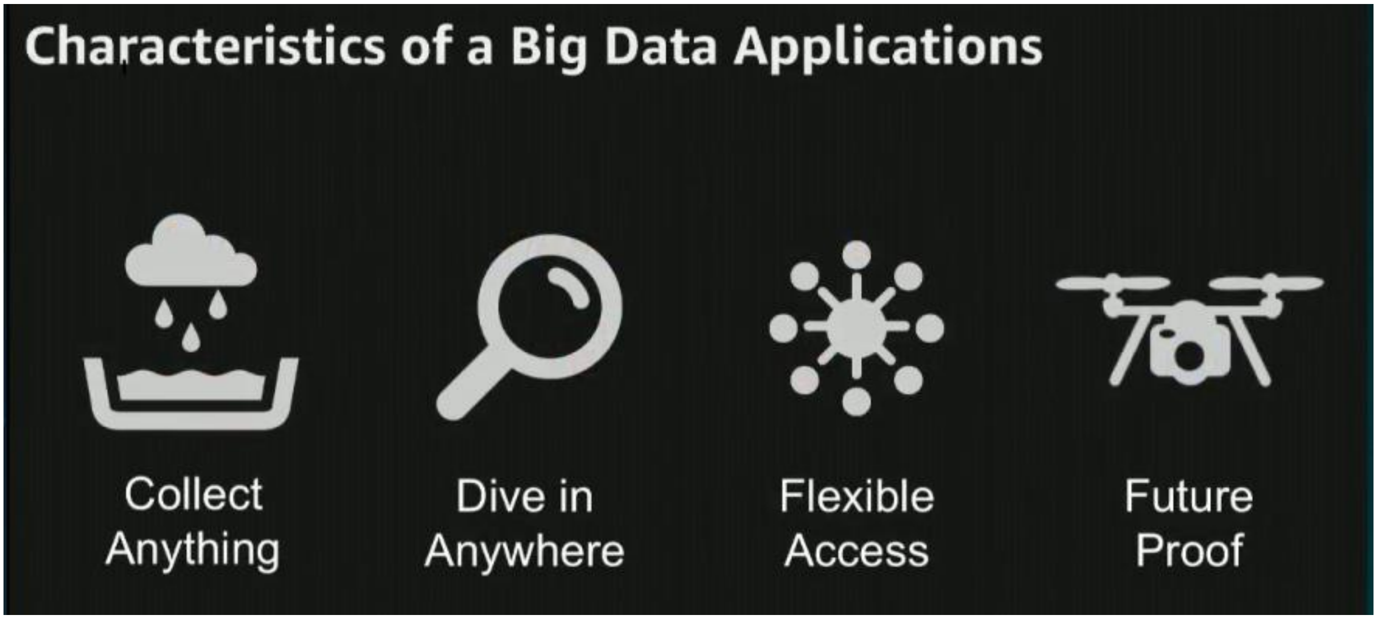
Data warehousing

Reporting

Real-time processing

Predictive analytics

Artificial Intelligence



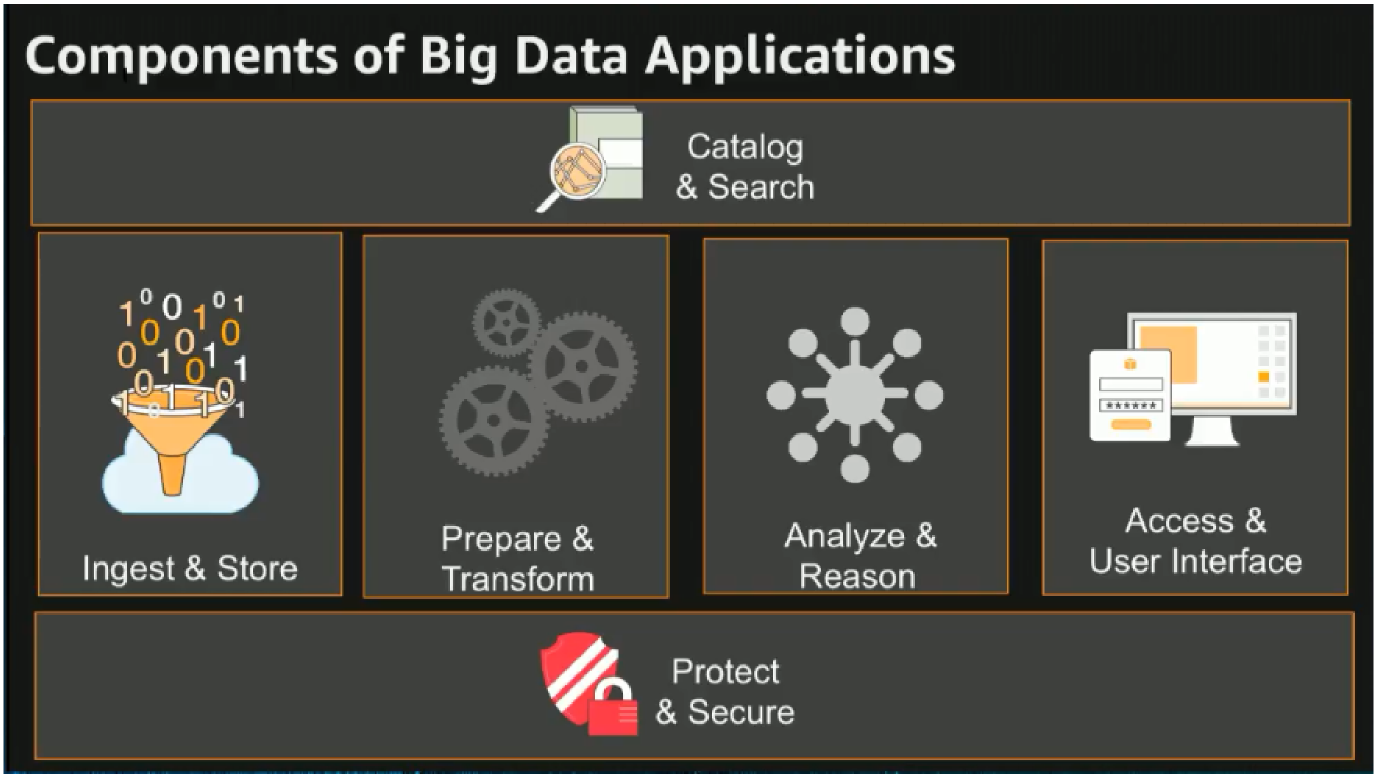
Characteristics of a Big Data Applications

Collect Anything

Dive in Anywhere

Flexible Access

Future Proof



Components of Big Data Applications

Catalog & Search

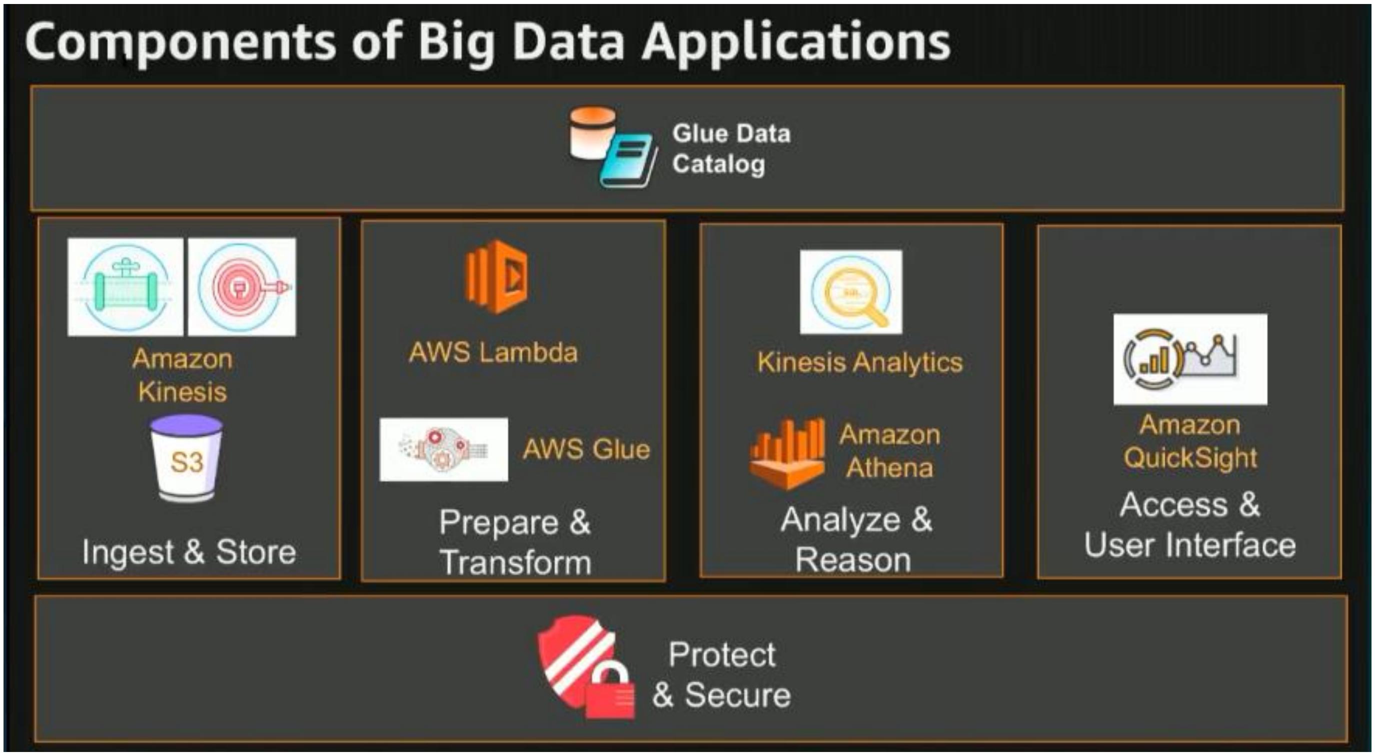
Ingest & Store

Prepare & Transform

Analyze & Reason

Access & User Interface

Protect & Secure



Components of Big Data Applications

Glue Data Catalog

Amazon Kinesis

S3

Ingest & Store

AWS Lambda

AWS Glue

Prepare & Transform

Kinesis Analytics

Amazon Athena

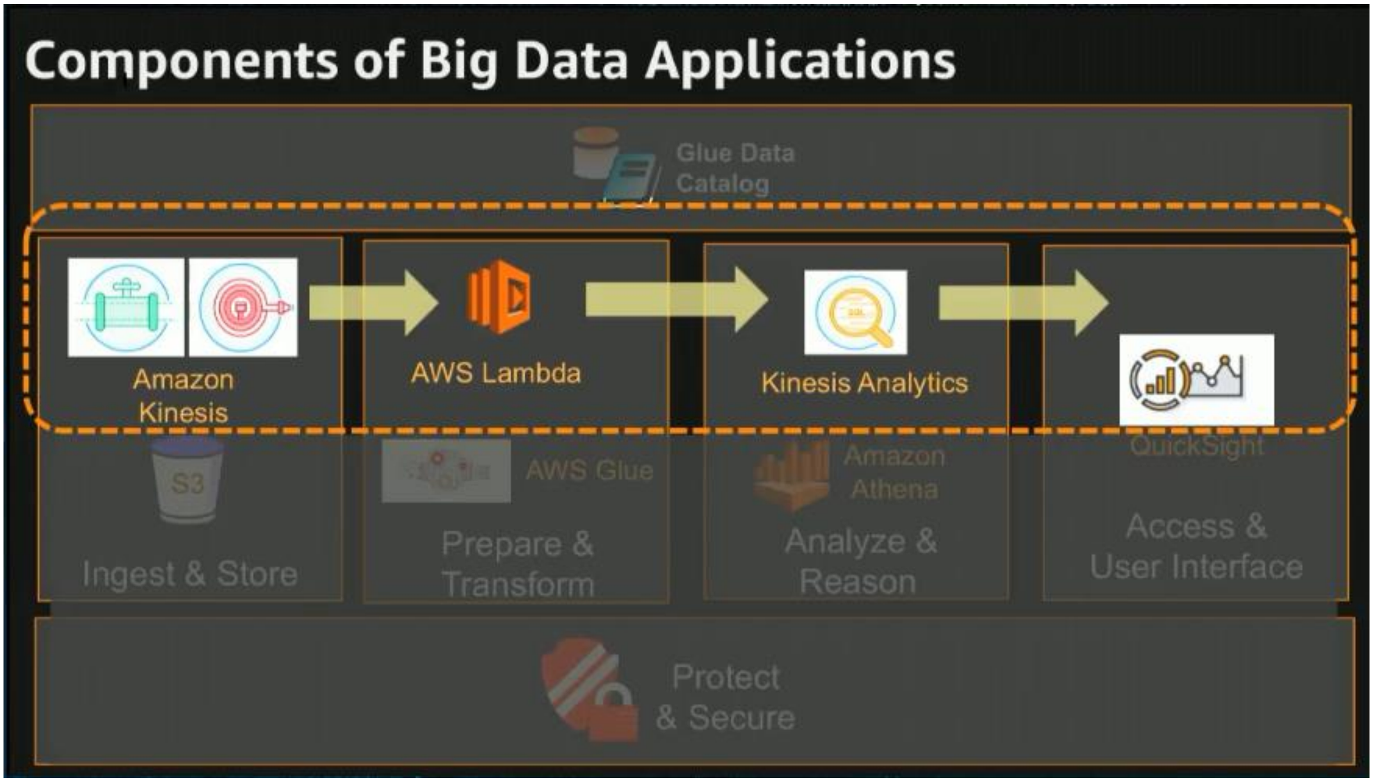
Analyze & Reason

Amazon QuickSight

Access & User Interface

Protect & Secure

AWS Glue has the capability to be a metadata catalog that helps you bring in and maintain new dataset. ***Lambda*** allows you to write transformation logic for your data as they come in, ***Glue*** also allows you to write transformations of your data also. ***Kinesis Analytics*** provides SQL based access on top of data streams while Athena provides SQL based access on top of your data stored on ***S3***.



Components of Big Data Applications

Glue Data Catalog

Amazon Kinesis

AWS Lambda

Kinesis Analytics

S3

AWS Glue

Amazon Athena

QuickSight

Ingest & Store

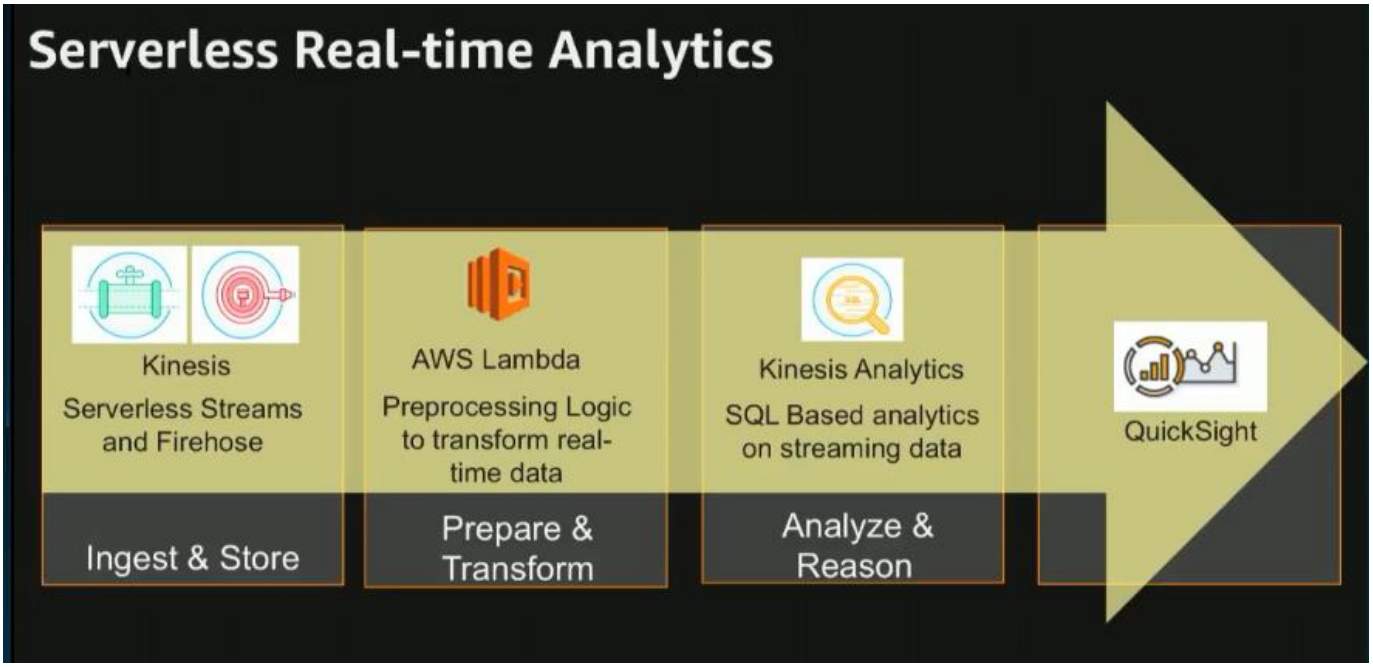
Prepare & Transform

Analyze & Reason

Access & User Interface

Protect & Secure

This is a very common real time analytical flow, it is very easy to setup to start capturing, transforming, and analyzing data



Serverless Real-time Analytics

Kinesis
Serverless Streams and Firehose

AWS Lambda
Preprocessing Logic to transform real-time data

Kinesis Analytics
SQL Based analytics on streaming data

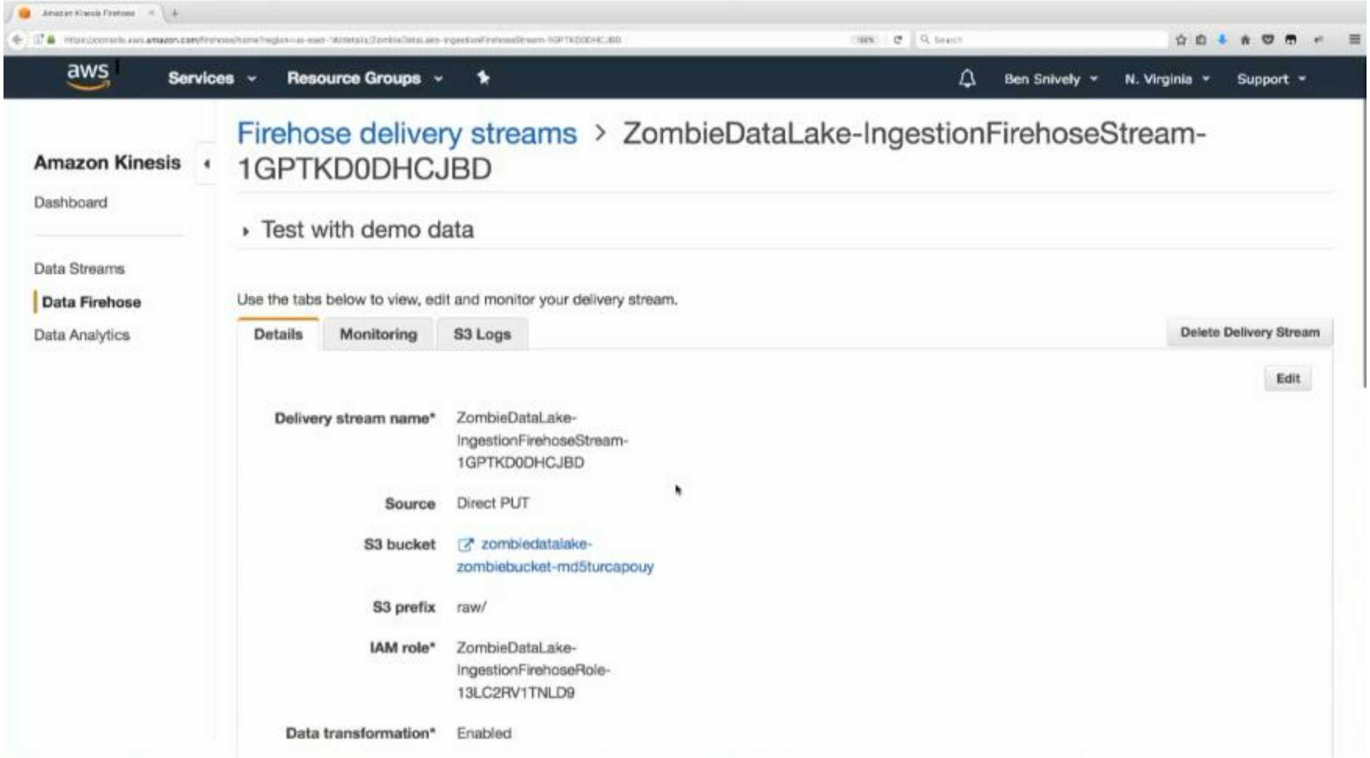
QuickSight

Ingest & Store

Prepare & Transform

Analyze & Reason

Demonstration

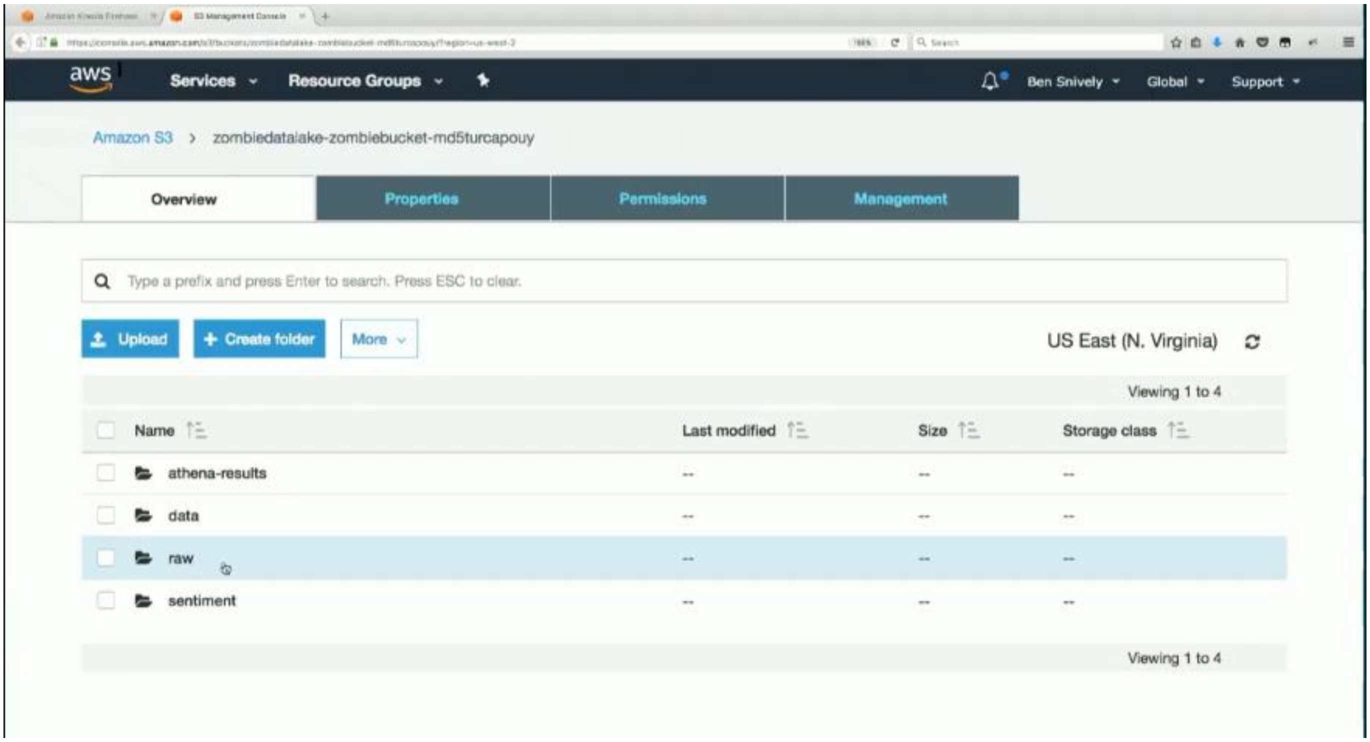


Firehose delivery streams > ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD

Use the tabs below to view, edit and monitor your delivery stream.

Delivery stream name*	ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD
Source	Direct PUT
S3 bucket	zombiedatalake-zombiebucket-md5turcapouy
S3 prefix	raw/
IAM role*	ZombieDataLake-IngestionFirehoseRole-13LC2RV1TNLD9
Data transformation*	Enabled

This is a Kinesis Firehose, we have specified an S3 bucket with a prefix ***raw/***. We want to capture app messages and analyzed the data



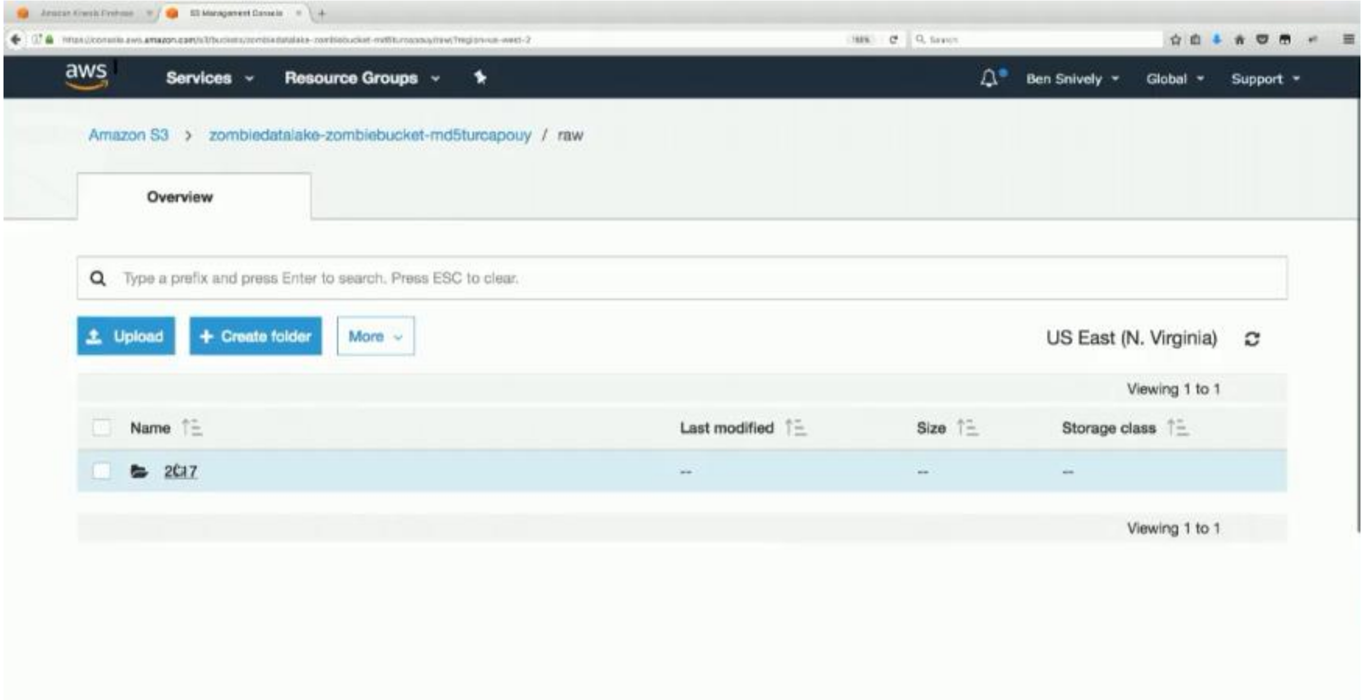
Amazon S3 > zombiedatalake-zombiebucket-md5turcapouy

US East (N. Virginia)

Viewing 1 to 4

Name	Last modified	Size	Storage class
athena-results	--	--	--
data	--	--	--
raw	--	--	--
sentiment	--	--	--

Viewing 1 to 4



Amazon S3 > zombiedatalake-zombiebucket-md5turcapouy / raw

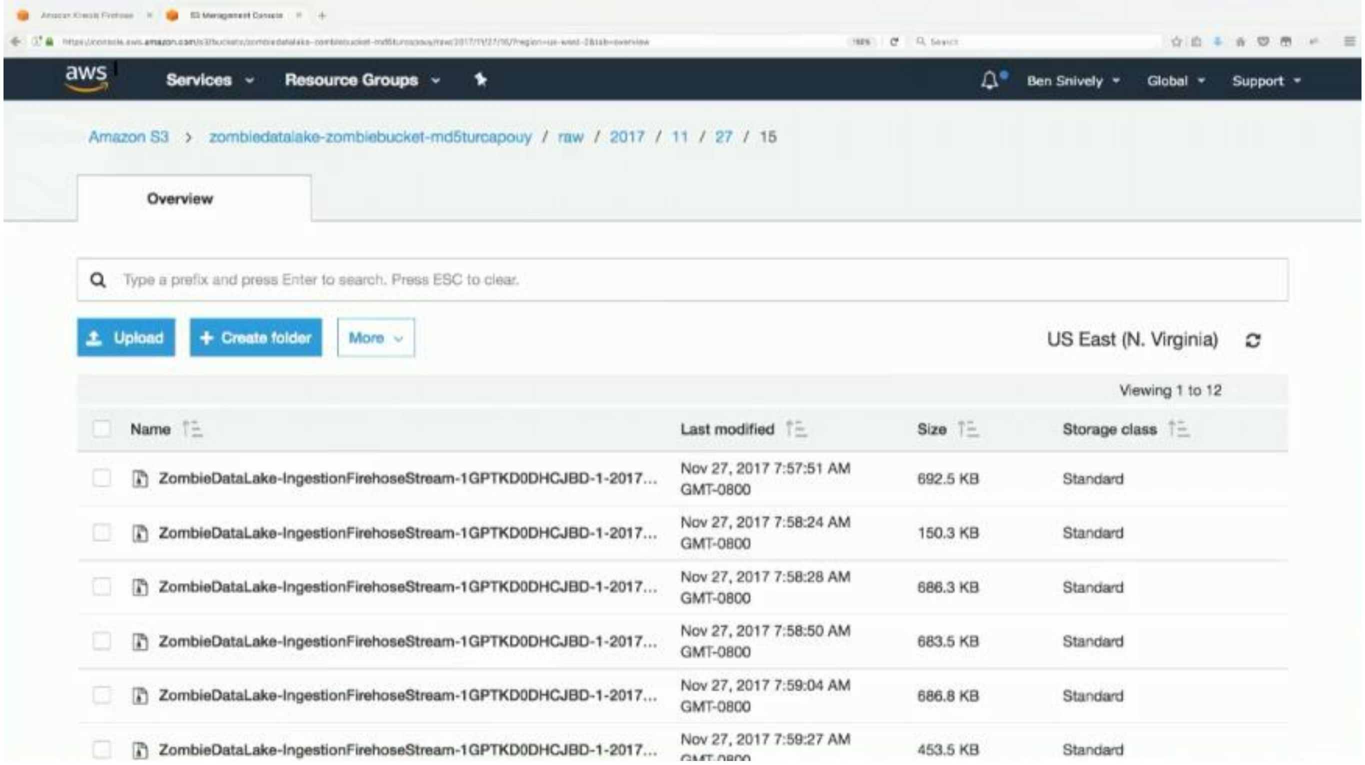
Type a prefix and press Enter to search. Press ESC to clear.

US East (N. Virginia)

Viewing 1 to 1

	Name	Last modified	Size	Storage class
□	2017	--	--	--

Viewing 1 to 1



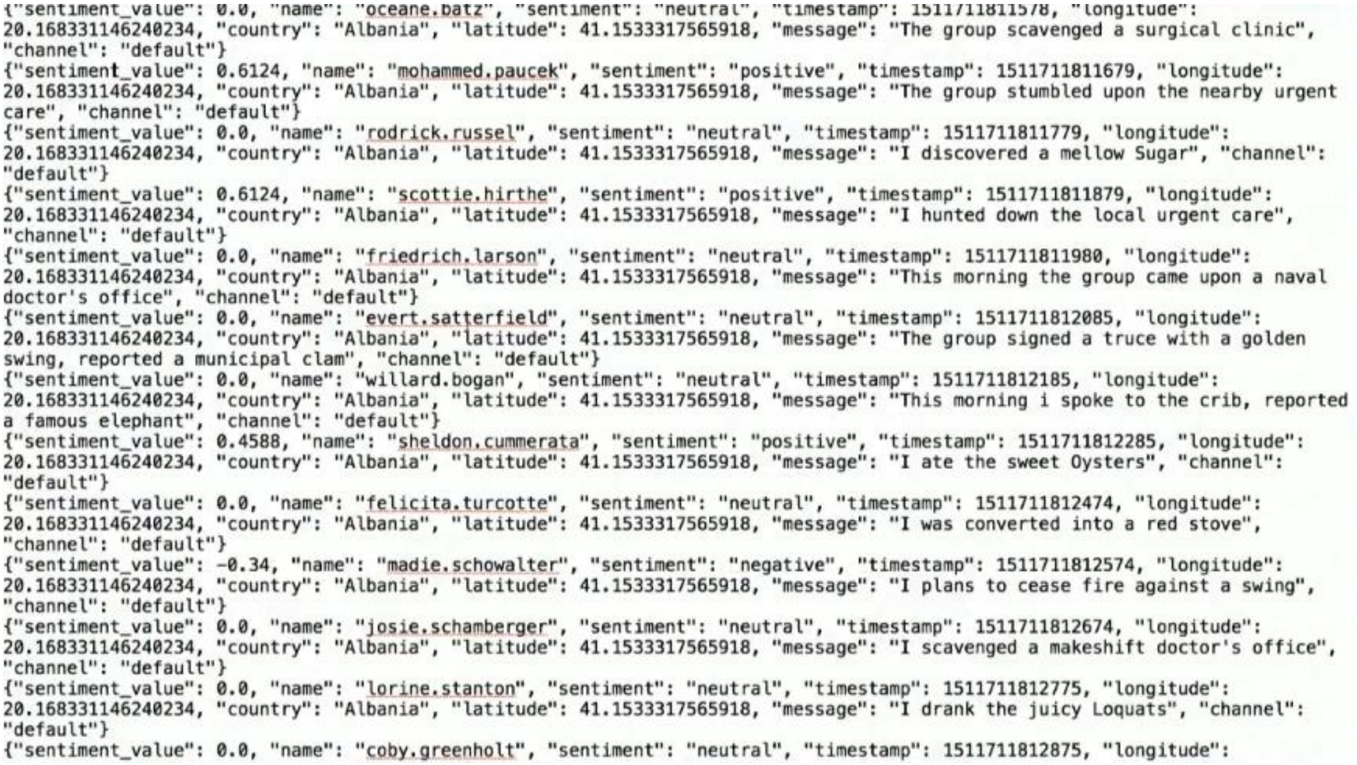
Amazon S3 > zombiedatalake-zombiebucket-md5turcapouy / raw / 2017 / 11 / 27 / 15

Type a prefix and press Enter to search. Press ESC to clear.

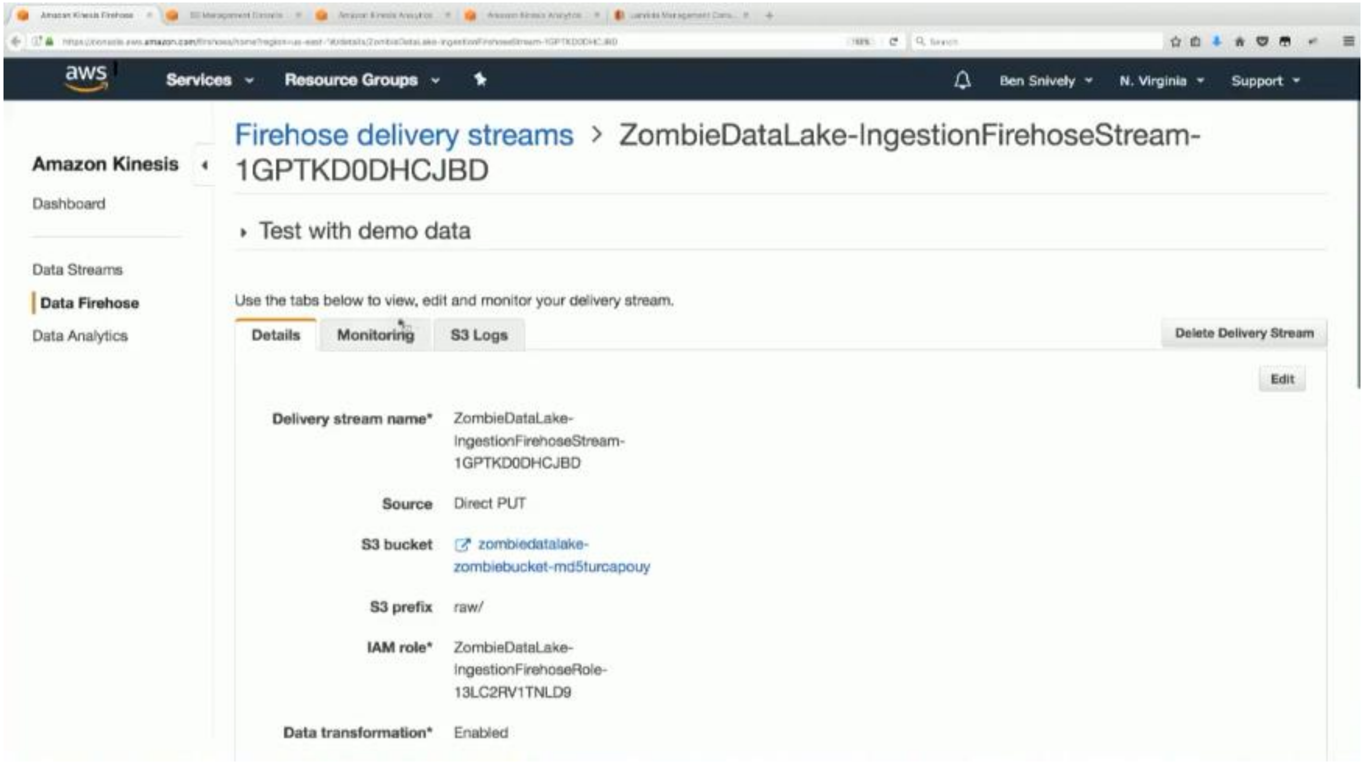
US East (N. Virginia)

Viewing 1 to 12

	Name	Last modified	Size	Storage class
□	ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD-1-2017...	Nov 27, 2017 7:57:51 AM GMT-0800	692.5 KB	Standard
□	ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD-1-2017...	Nov 27, 2017 7:58:24 AM GMT-0800	150.3 KB	Standard
□	ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD-1-2017...	Nov 27, 2017 7:58:28 AM GMT-0800	686.3 KB	Standard
□	ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD-1-2017...	Nov 27, 2017 7:58:50 AM GMT-0800	683.5 KB	Standard
□	ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD-1-2017...	Nov 27, 2017 7:59:04 AM GMT-0800	686.8 KB	Standard
□	ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD-1-2017...	Nov 27, 2017 7:59:27 AM GMT-0800	453.5 KB	Standard



```
{"sentiment_value": 0.0, "name": "oceane.batz", "sentiment": "neutral", "timestamp": 1511711811578, "longitude":
20.168331146240234, "country": "Albania", "latitude": 41.1533317565918, "message": "The group scavenged a surgical clinic",
"channel": "default"}
{"sentiment_value": 0.6124, "name": "mohammed.paucek", "sentiment": "positive", "timestamp": 1511711811679, "longitude":
20.168331146240234, "country": "Albania", "latitude": 41.1533317565918, "message": "The group stumbled upon the nearby urgent
care", "channel": "default"}
{"sentiment_value": 0.0, "name": "rodrick.russel", "sentiment": "neutral", "timestamp": 1511711811779, "longitude":
20.168331146240234, "country": "Albania", "latitude": 41.1533317565918, "message": "I discovered a mellow Sugar", "channel":
"default"}
{"sentiment_value": 0.6124, "name": "scottie.hirthe", "sentiment": "positive", "timestamp": 1511711811879, "longitude":
20.168331146240234, "country": "Albania", "latitude": 41.1533317565918, "message": "I hunted down the local urgent care",
"channel": "default"}
{"sentiment_value": 0.0, "name": "friedrich.larson", "sentiment": "neutral", "timestamp": 1511711811980, "longitude":
20.168331146240234, "country": "Albania", "latitude": 41.1533317565918, "message": "This morning the group came upon a naval
doctor's office", "channel": "default"}
{"sentiment_value": 0.0, "name": "evert.satterfield", "sentiment": "neutral", "timestamp": 1511711812085, "longitude":
20.168331146240234, "country": "Albania", "latitude": 41.1533317565918, "message": "The group signed a truce with a golden
swing, reported a municipal clam", "channel": "default"}
{"sentiment_value": 0.0, "name": "willard.bogan", "sentiment": "neutral", "timestamp": 1511711812185, "longitude":
20.168331146240234, "country": "Albania", "latitude": 41.1533317565918, "message": "This morning i spoke to the crib, reported
a famous elephant", "channel": "default"}
{"sentiment_value": 0.4588, "name": "sheldon.cummerata", "sentiment": "positive", "timestamp": 1511711812285, "longitude":
20.168331146240234, "country": "Albania", "latitude": 41.1533317565918, "message": "I ate the sweet Oysters", "channel":
"default"}
{"sentiment_value": 0.0, "name": "felicita.turcotte", "sentiment": "neutral", "timestamp": 1511711812474, "longitude":
20.168331146240234, "country": "Albania", "latitude": 41.1533317565918, "message": "I was converted into a red stove",
"channel": "default"}
{"sentiment_value": -0.34, "name": "madie.schowalter", "sentiment": "negative", "timestamp": 1511711812574, "longitude":
20.168331146240234, "country": "Albania", "latitude": 41.1533317565918, "message": "I plans to cease fire against a swing",
"channel": "default"}
{"sentiment_value": 0.0, "name": "josie.schamberger", "sentiment": "neutral", "timestamp": 1511711812674, "longitude":
20.168331146240234, "country": "Albania", "latitude": 41.1533317565918, "message": "I scavenged a makeshift doctor's office",
"channel": "default"}
{"sentiment_value": 0.0, "name": "lorine.stanton", "sentiment": "neutral", "timestamp": 1511711812775, "longitude":
20.168331146240234, "country": "Albania", "latitude": 41.1533317565918, "message": "I drank the juicy Loquats", "channel":
"default"}
{"sentiment_value": 0.0, "name": "coby.greenholt", "sentiment": "neutral", "timestamp": 1511711812875, "longitude":
```



Firehose delivery streams > ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD

▸ Test with demo data

Use the tabs below to view, edit and monitor your delivery stream.

Delivery stream name* ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD

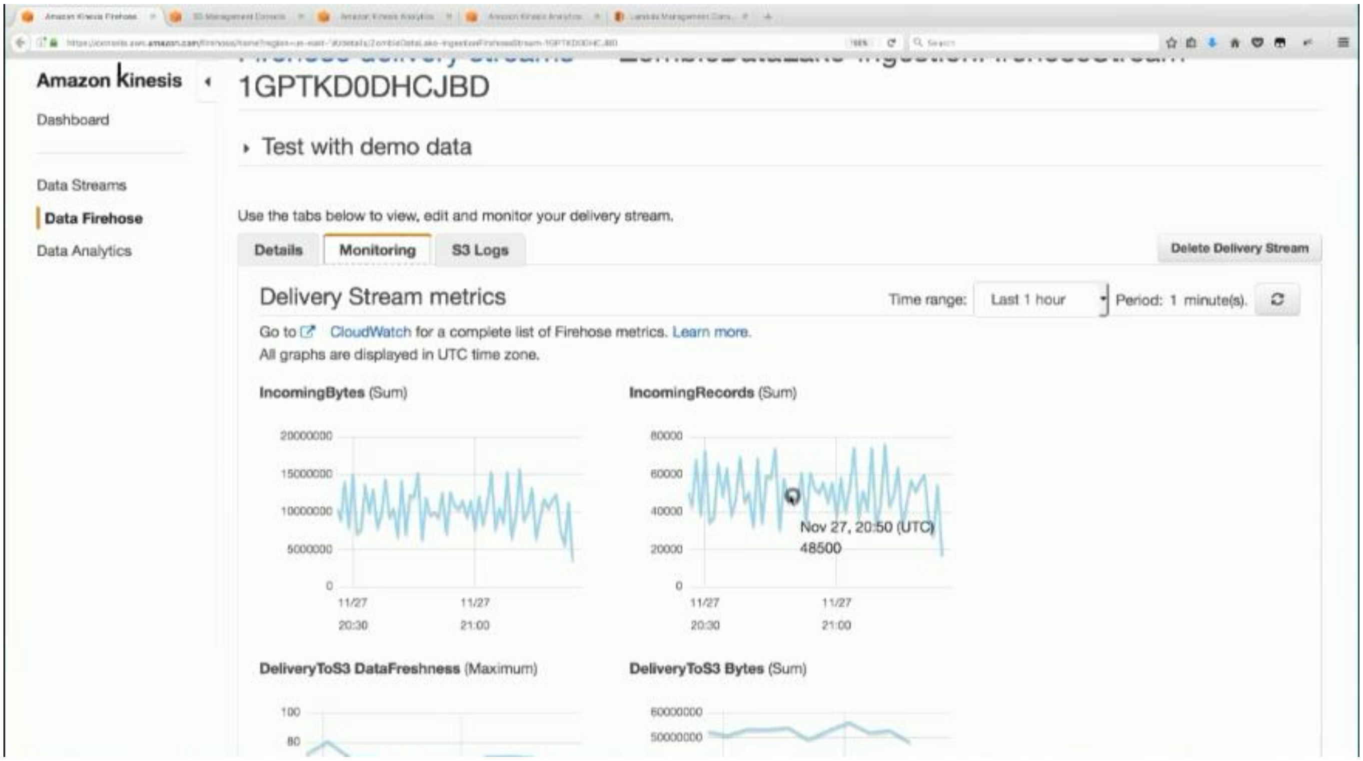
Source Direct PUT

S3 bucket zombiedatalake-zombiebucket-md5turcapouy

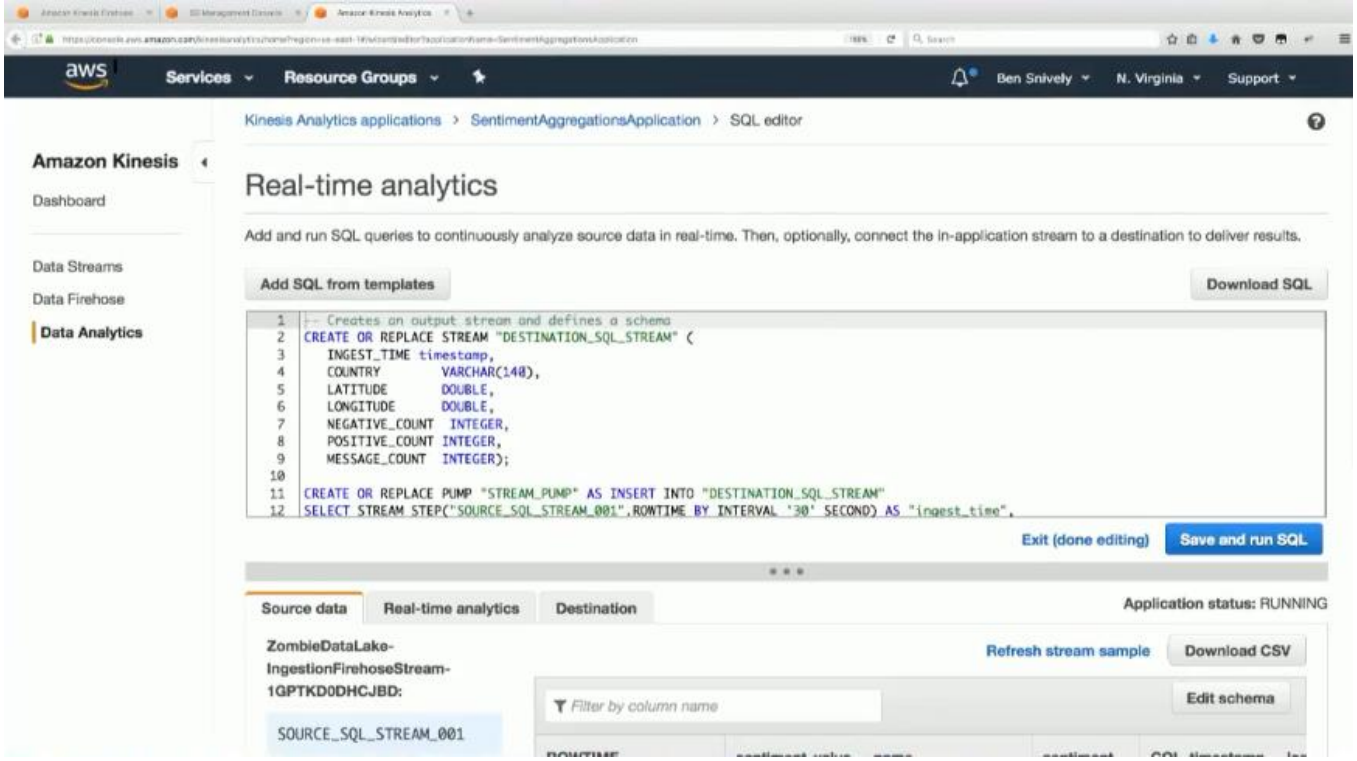
S3 prefix raw/

IAM role* ZombieDataLake-IngestionFirehoseRole-13LC2RV1TNLD9

Data transformation* Enabled



We can easily query this data using Athena or Kinesis Analytics as shown below



Real-time analytics

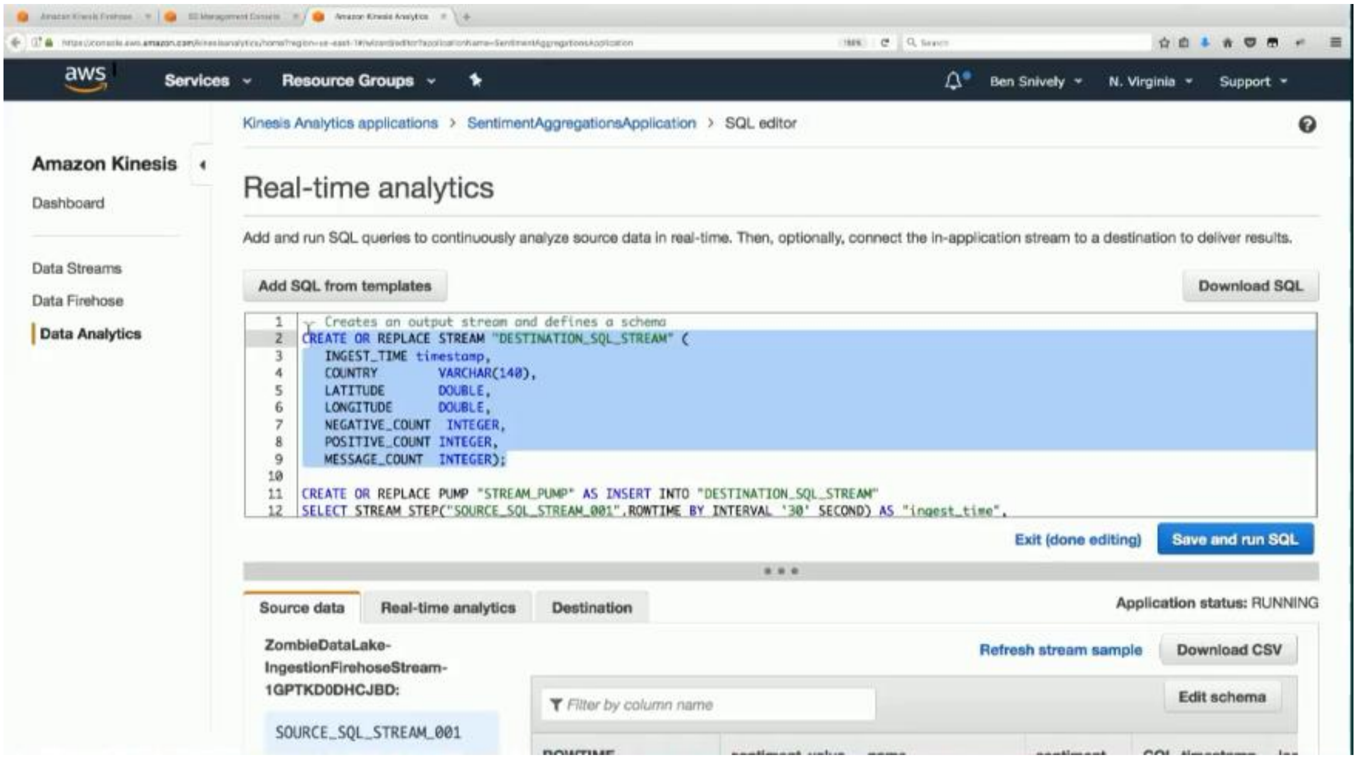
Add and run SQL queries to continuously analyze source data in real-time. Then, optionally, connect the in-application stream to a destination to deliver results.

```
-- Creates an output stream and defines a schema
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
   INGEST_TIME timestamp,
   COUNTRY        VARCHAR(140),
   LATITUDE       DOUBLE,
   LONGITUDE      DOUBLE,
   NEGATIVE_COUNT  INTEGER,
   POSITIVE_COUNT INTEGER,
   MESSAGE_COUNT  INTEGER);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT STREAM STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '30' SECOND) AS "ingest_time",
```

Application status: RUNNING

ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD:

We write a SQL query, we have a SQL statement that does our real time analytics and start getting results



Kinesis Analytics applications > SentimentAggregationsApplication > SQL editor

Real-time analytics

Add and run SQL queries to continuously analyze source data in real-time. Then, optionally, connect the in-application stream to a destination to deliver results.

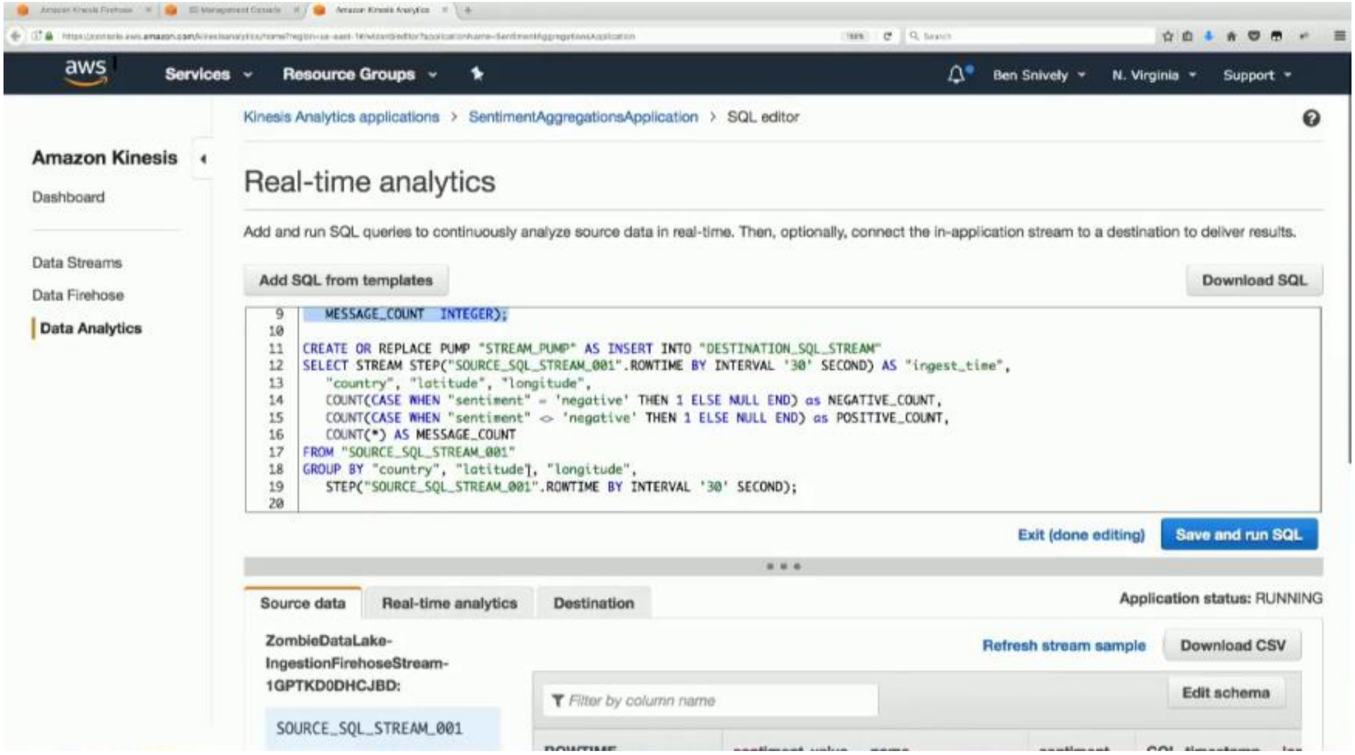
```
-- Creates an output stream and defines a schema
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
   INGEST_TIME timestamp,
   COUNTRY        VARCHAR(140),
   LATITUDE       DOUBLE,
   LONGITUDE      DOUBLE,
   NEGATIVE_COUNT  INTEGER,
   POSITIVE_COUNT INTEGER,
   MESSAGE_COUNT  INTEGER);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT STREAM STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '30' SECOND) AS "ingest_time",
```

Application status: RUNNING

ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD:

SOURCE_SQL_STREAM_001

This defines the output using a sliding window



Kinesis Analytics applications > SentimentAggregationsApplication > SQL editor

Real-time analytics

Add and run SQL queries to continuously analyze source data in real-time. Then, optionally, connect the in-application stream to a destination to deliver results.

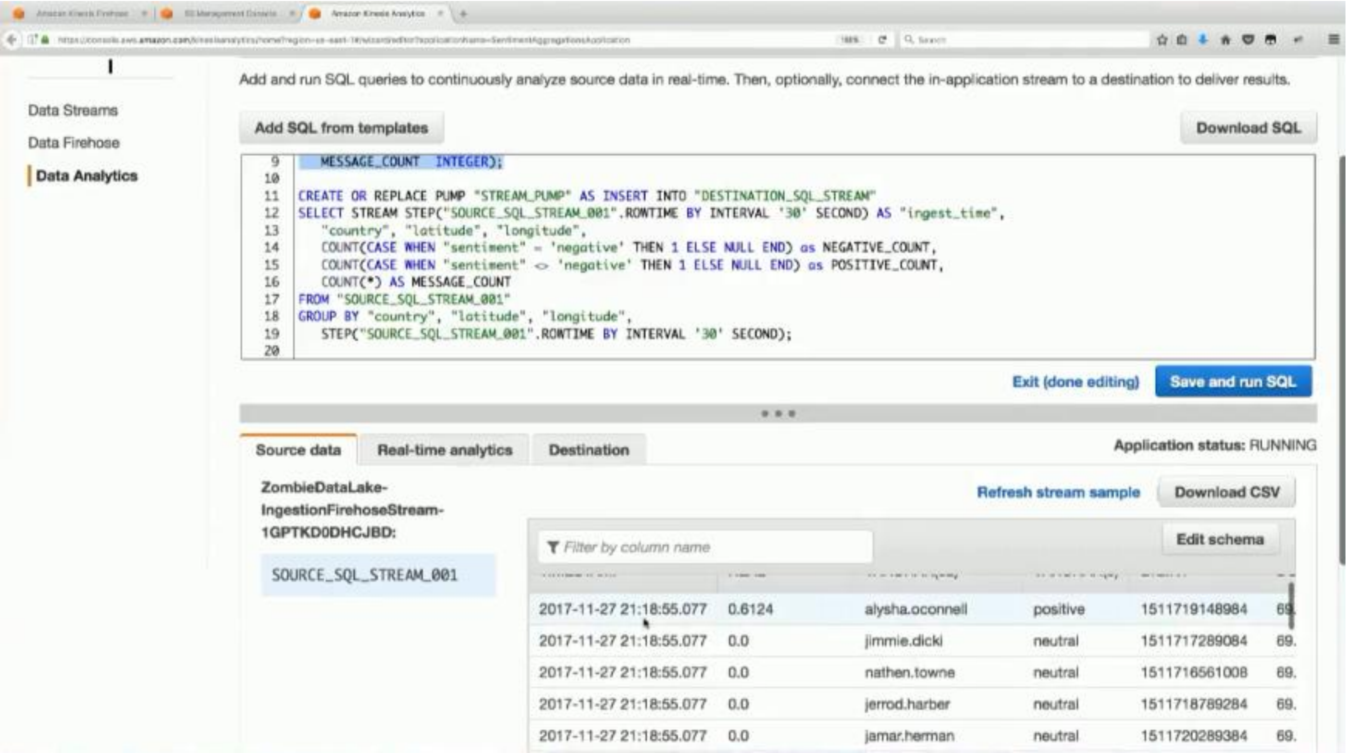
```
   MESSAGE_COUNT  INTEGER);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT STREAM STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '30' SECOND) AS "ingest_time",
   "country", "latitude", "longitude",
   COUNT(CASE WHEN "sentiment" = 'negative' THEN 1 ELSE NULL END) as NEGATIVE_COUNT,
   COUNT(CASE WHEN "sentiment" <> 'negative' THEN 1 ELSE NULL END) as POSITIVE_COUNT,
   COUNT(*) AS MESSAGE_COUNT
FROM "SOURCE_SQL_STREAM_001"
GROUP BY "country", "latitude", "longitude",
   STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '30' SECOND);
```

Application status: RUNNING

ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD:

SOURCE_SQL_STREAM_001

We are now generating the data into the stream, this is doing a 30 second window of data



Add and run SQL queries to continuously analyze source data in real-time. Then, optionally, connect the in-application stream to a destination to deliver results.

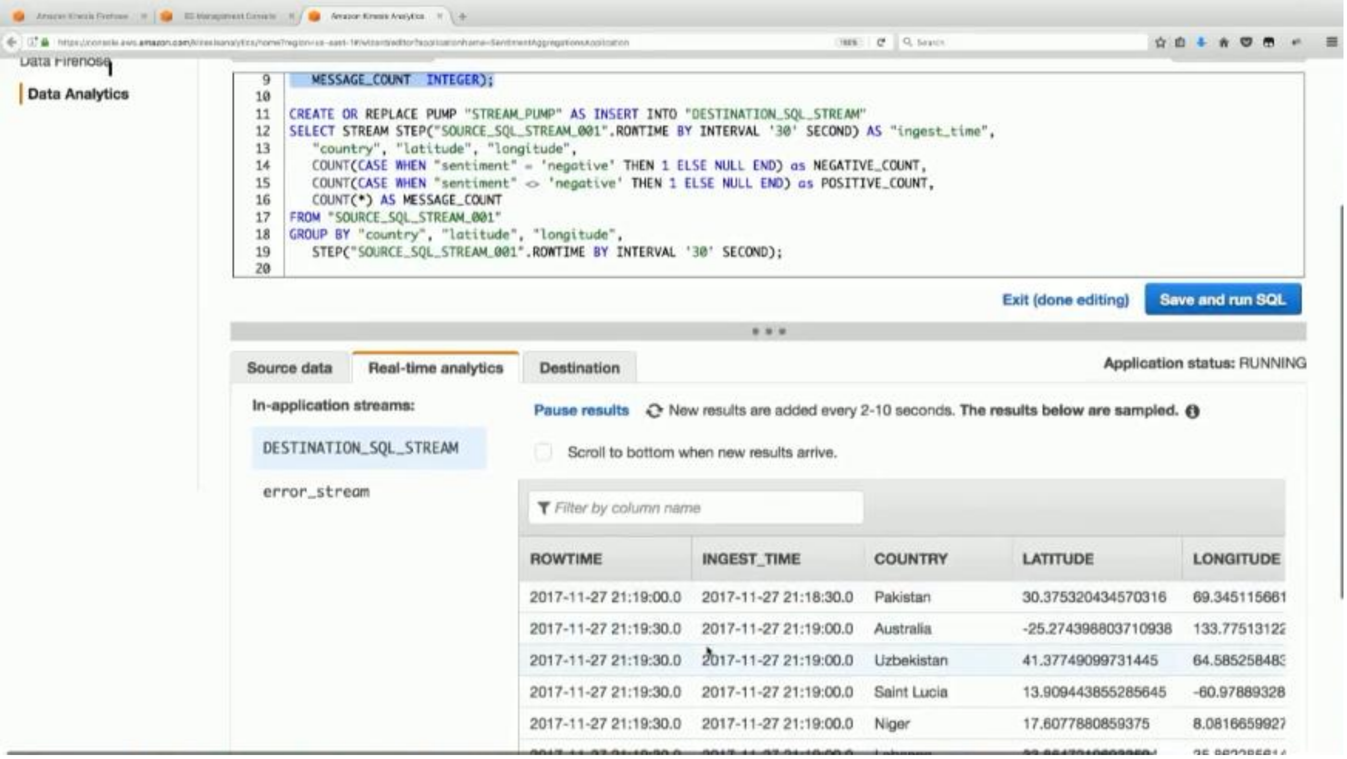
```
   MESSAGE_COUNT  INTEGER);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT STREAM STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '30' SECOND) AS "ingest_time",
   "country", "latitude", "longitude",
   COUNT(CASE WHEN "sentiment" = 'negative' THEN 1 ELSE NULL END) as NEGATIVE_COUNT,
   COUNT(CASE WHEN "sentiment" <> 'negative' THEN 1 ELSE NULL END) as POSITIVE_COUNT,
   COUNT(*) AS MESSAGE_COUNT
FROM "SOURCE_SQL_STREAM_001"
GROUP BY "country", "latitude", "longitude",
   STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '30' SECOND);
```

Application status: RUNNING

ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD:

SOURCE_SQL_STREAM_001

2017-11-27 21:18:55.077	0.6124	alysha.oconnell	positive	1511719148984	69
2017-11-27 21:18:55.077	0.0	jimmie.dicki	neutral	1511717289084	69.
2017-11-27 21:18:55.077	0.0	nathen.towne	neutral	1511716561008	69.
2017-11-27 21:18:55.077	0.0	jerrod.harber	neutral	1511718789284	69.
2017-11-27 21:18:55.077	0.0	jamar.herman	neutral	1511720289384	69.



```
   MESSAGE_COUNT  INTEGER);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT STREAM STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '30' SECOND) AS "ingest_time",
   "country", "latitude", "longitude",
   COUNT(CASE WHEN "sentiment" = 'negative' THEN 1 ELSE NULL END) as NEGATIVE_COUNT,
   COUNT(CASE WHEN "sentiment" <> 'negative' THEN 1 ELSE NULL END) as POSITIVE_COUNT,
   COUNT(*) AS MESSAGE_COUNT
FROM "SOURCE_SQL_STREAM_001"
GROUP BY "country", "latitude", "longitude",
   STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '30' SECOND);
```

Application status: RUNNING

In-application streams:

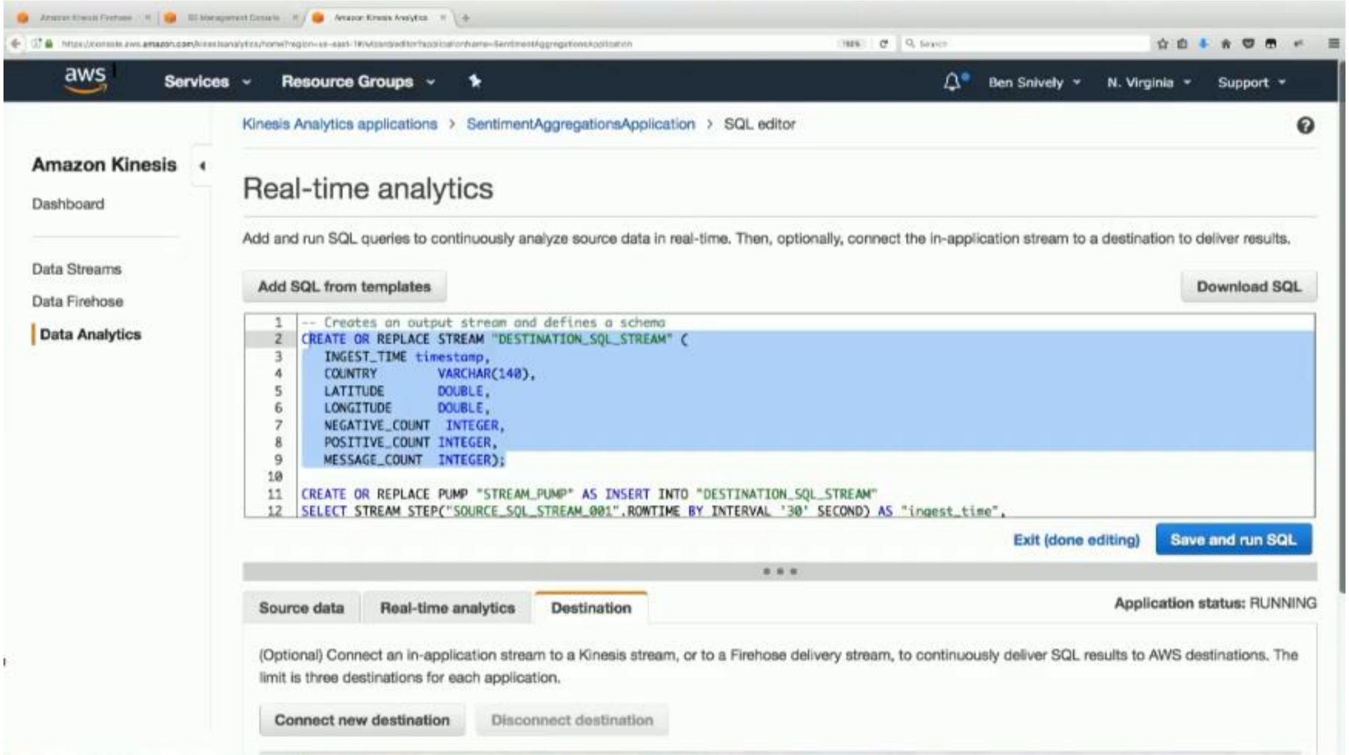
DESTINATION_SQL_STREAM

error_stream

New results are added every 2-10 seconds. **The results below are sampled.**

□ Scroll to bottom when new results arrive.

ROWTIME	INGEST_TIME	COUNTRY	LATITUDE	LONGITUDE
2017-11-27 21:19:00.0	2017-11-27 21:18:30.0	Pakistan	30.375320434570316	69.345115661
2017-11-27 21:19:30.0	2017-11-27 21:19:00.0	Australia	-25.274398803710938	133.77513122
2017-11-27 21:19:30.0	2017-11-27 21:19:00.0	Uzbekistan	41.37749099731445	64.58525848
2017-11-27 21:19:30.0	2017-11-27 21:19:00.0	Saint Lucia	13.909443855285645	-60.97889328
2017-11-27 21:19:30.0	2017-11-27 21:19:00.0	Niger	17.6077880859375	8.0816659927



Real-time analytics

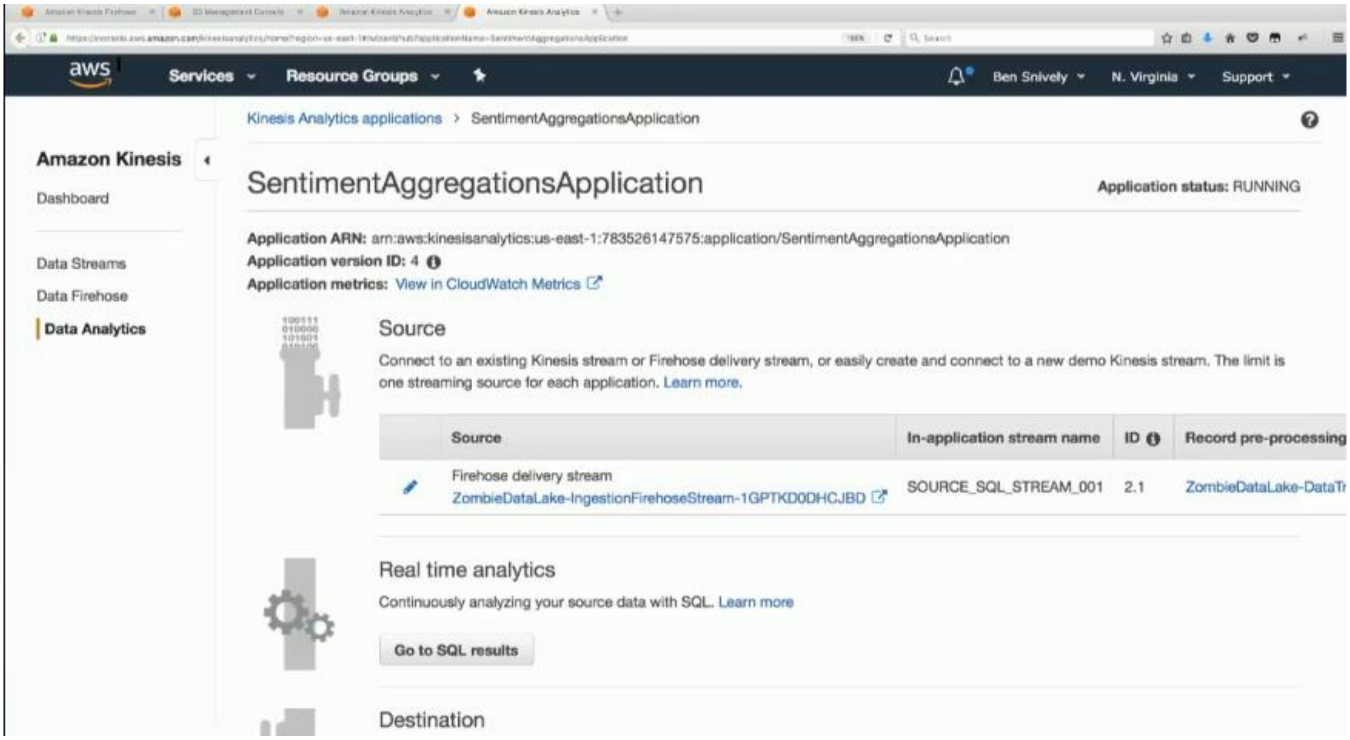
Add and run SQL queries to continuously analyze source data in real-time. Then, optionally, connect the in-application stream to a destination to deliver results.

```
-- Creates an output stream and defines a schema
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
   INGEST_TIME timestamp,
   COUNTRY        VARCHAR(140),
   LATITUDE       DOUBLE,
   LONGITUDE      DOUBLE,
   NEGATIVE_COUNT  INTEGER,
   POSITIVE_COUNT INTEGER,
   MESSAGE_COUNT  INTEGER);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT STREAM STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '30' SECOND) AS "ingest_time",
```

Application status: RUNNING

(Optional) Connect an in-application stream to a Kinesis stream, or to a Firehose delivery stream, to continuously deliver SQL results to AWS destinations. The limit is three destinations for each application.

We can specify a destination like S3 for the analyzed result



SentimentAggregationsApplication

Application status: RUNNING

Application ARN: arn:aws:kinesisanalytics:us-east-1:783526147575:application/SentimentAggregationsApplication
Application version ID: 4
Application metrics: View in CloudWatch Metrics

Source

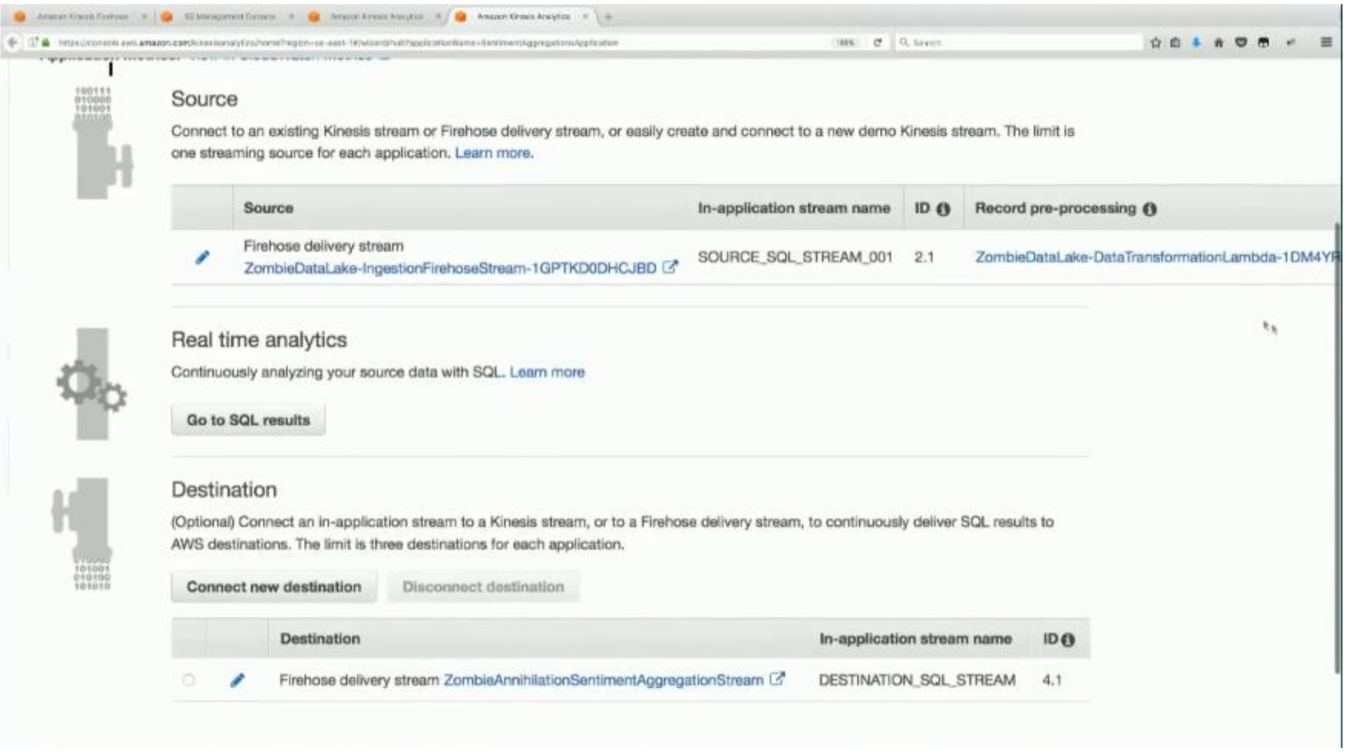
Connect to an existing Kinesis stream or Firehose delivery stream, or easily create and connect to a new demo Kinesis stream. The limit is one streaming source for each application. Learn more.

Source	In-application stream name	ID	Record pre-processing
Firehose delivery stream ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD	SOURCE_SQL_STREAM_001	2.1	ZombieDataLake-DataTr

Real time analytics

Continuously analyzing your source data with SQL. Learn more

Destination



Source

Connect to an existing Kinesis stream or Firehose delivery stream, or easily create and connect to a new demo Kinesis stream. The limit is one streaming source for each application. Learn more.

	Source	In-application stream name	ID	Record pre-processing
✎	Firehose delivery stream ZombieDataLake-IngestionFirehoseStream-1GPTKD0DHCJBD	SOURCE_SQL_STREAM_001	2.1	ZombieDataLake-DataTransformationLambda-1DM4YR

Real time analytics

Continuously analyzing your source data with SQL. Learn more

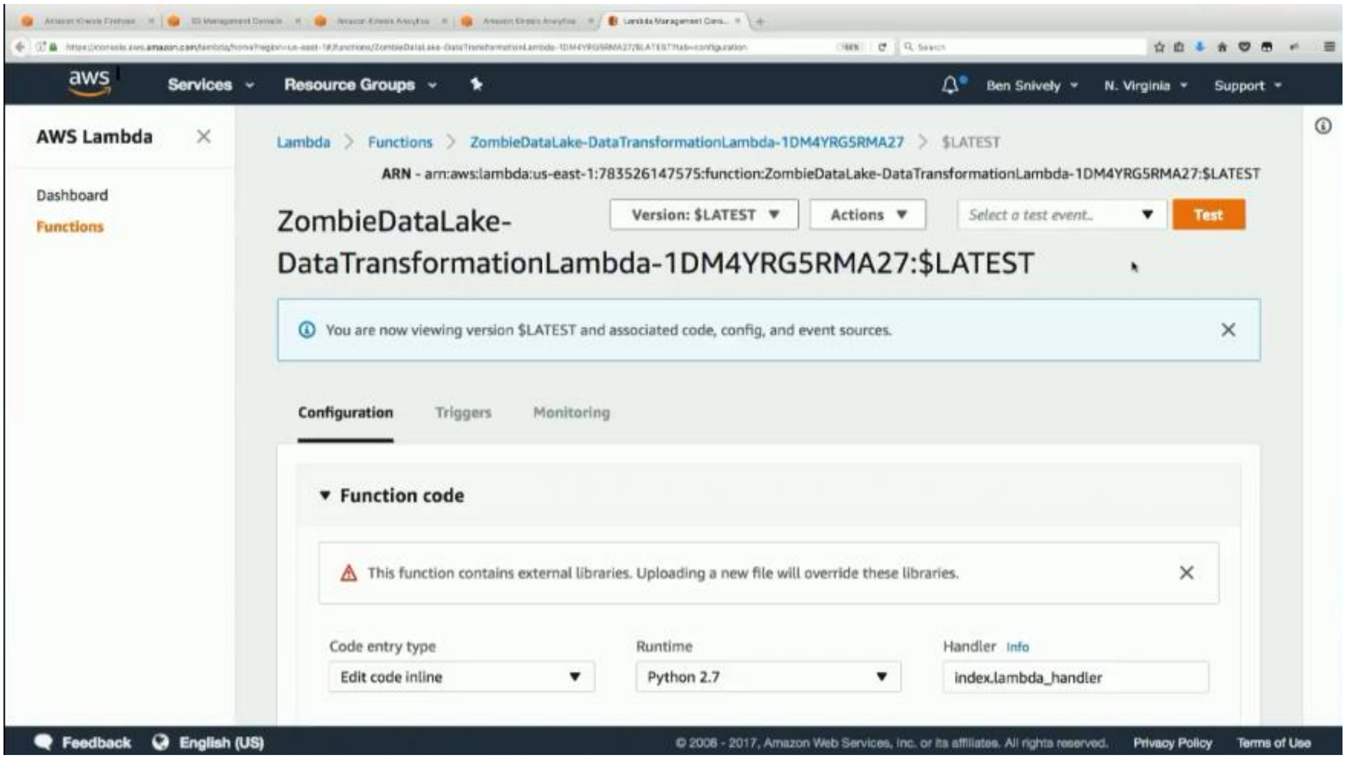
Go to SQL results

Destination

(Optional) Connect an in-application stream to a Kinesis stream, or to a Firehose delivery stream, to continuously deliver SQL results to AWS destinations. The limit is three destinations for each application.

Connect new destination | Disconnect destination

		Destination	In-application stream name	ID
○	✎	Firehose delivery stream ZombieAnnihilationSentimentAggregationStream	DESTINATION_SQL_STREAM	4.1



Lambda > Functions > ZombieDataLake-DataTransformationLambda-1DM4YRG5RMA27 > $LATEST

ARN - arn:aws:lambda:us-east-1:783526147575:function:ZombieDataLake-DataTransformationLambda-1DM4YRG5RMA27:$LATEST

ZombieDataLake-DataTransformationLambda-1DM4YRG5RMA27:$LATEST

Version: $LATEST ▼ | Actions ▼ | Select a test event.. ▼ | Test

ⓘ You are now viewing version $LATEST and associated code, config, and event sources.

Configuration | Triggers | Monitoring

▾ Function code

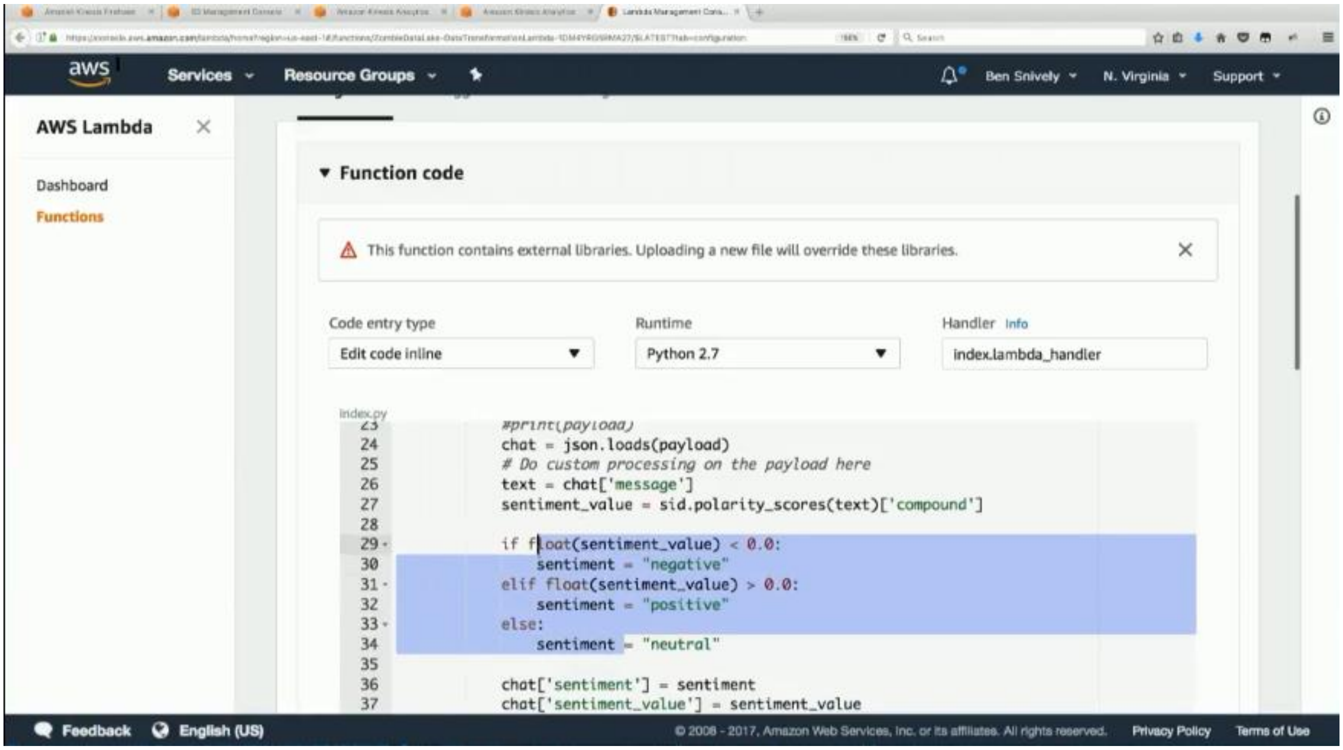
⚠ This function contains external libraries. Uploading a new file will override these libraries.

Code entry type: Edit code inline ▼

Runtime: Python 2.7 ▼

Handler Info: index.lambda_handler

Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use



This function contains external libraries. Uploading a new file will override these libraries.

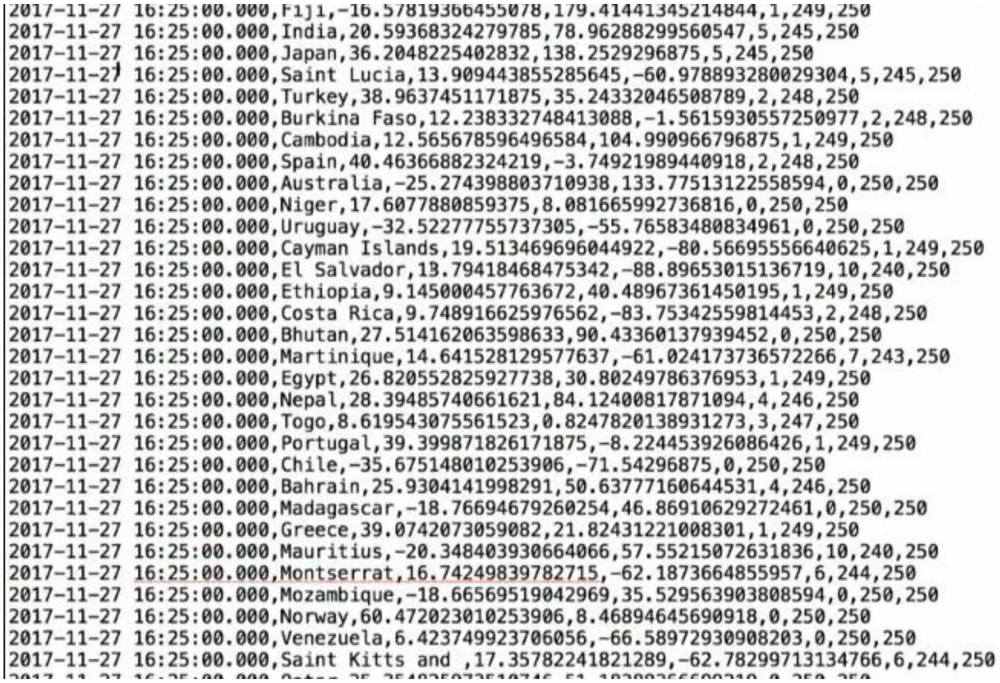
Code entry type: Edit code inline

Runtime: Python 2.7

Handler: index.lambda_handler

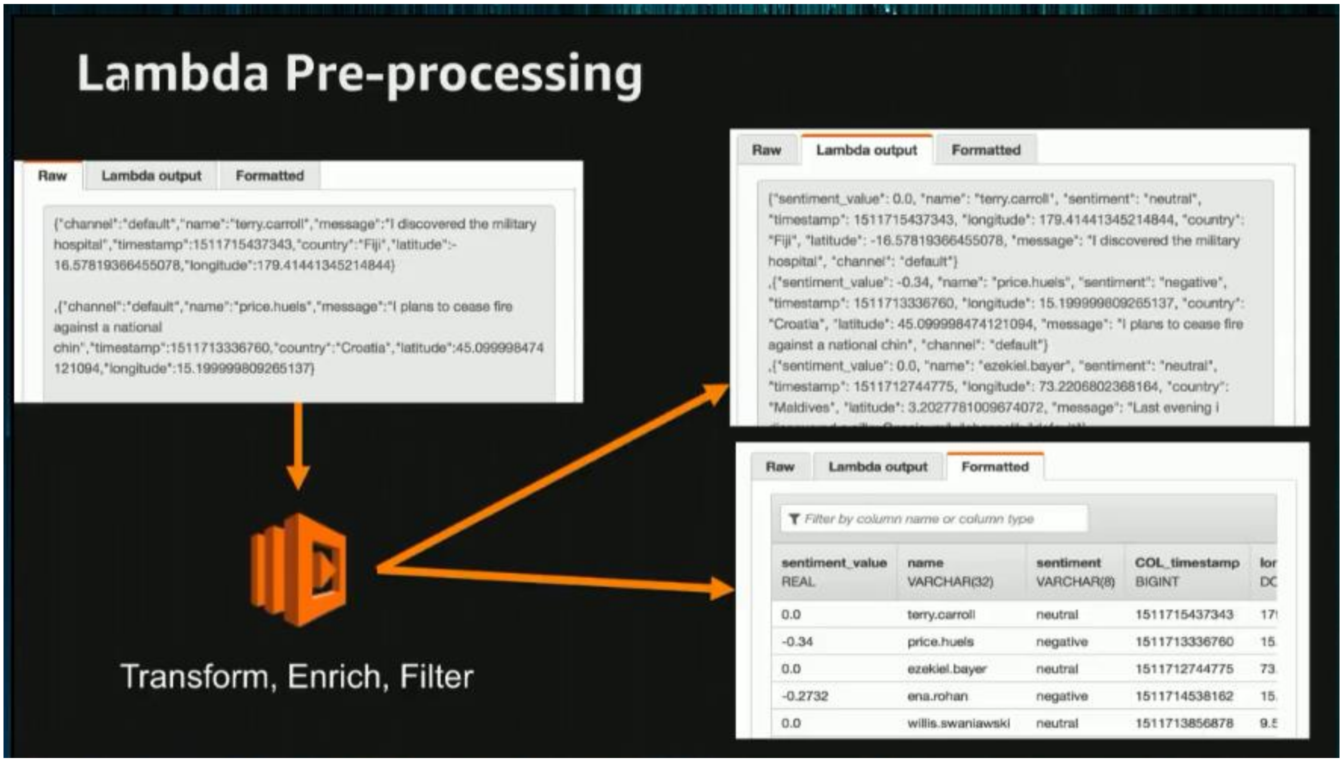
```
        #print(payload)
        chat = json.loads(payload)
        # Do custom processing on the payload here
        text = chat['message']
        sentiment_value = sid.polarity_scores(text)['compound']

        if float(sentiment_value) < 0.0:
            sentiment = "negative"
        elif float(sentiment_value) > 0.0:
            sentiment = "positive"
        else:
            sentiment = "neutral"

        chat['sentiment'] = sentiment
        chat['sentiment_value'] = sentiment_value
```

We can run some custom lambda code to transform or enrich the data coming into the stream before storing it



```
2017-11-27 16:25:00.000,Fiji,-16.578193664455078,179.41441345214844,1,249,250
2017-11-27 16:25:00.000,India,20.59368324279785,78.96288299560547,5,245,250
2017-11-27 16:25:00.000,Japan,36.2048225402832,138.2529296875,5,245,250
2017-11-27 16:25:00.000,Saint Lucia,13.909443855285645,-60.978893280029304,5,245,250
2017-11-27 16:25:00.000,Turkey,38.9637451171875,35.24332046508789,2,248,250
2017-11-27 16:25:00.000,Burkina Faso,12.238332748413088,-1.5615930557250977,2,248,250
2017-11-27 16:25:00.000,Cambodia,12.565678596496584,104.990966796875,1,249,250
2017-11-27 16:25:00.000,Spain,40.46366882324219,-3.74921989440918,2,248,250
2017-11-27 16:25:00.000,Australia,-25.274398803710938,133.77513122558594,0,250,250
2017-11-27 16:25:00.000,Niger,17.6077880859375,8.081665992736816,0,250,250
2017-11-27 16:25:00.000,Uruguay,-32.52277755737305,-55.76583480834961,0,250,250
2017-11-27 16:25:00.000,Cayman Islands,19.513469696044922,-80.56695556640625,1,249,250
2017-11-27 16:25:00.000,El Salvador,13.79418468475342,-88.89653015136719,10,240,250
2017-11-27 16:25:00.000,Ethiopia,9.145000457763672,40.48967361450195,1,249,250
2017-11-27 16:25:00.000,Costa Rica,9.748916625976562,-83.75342559814453,2,248,250
2017-11-27 16:25:00.000,Bhutan,27.514162063598633,90.43360137939452,0,250,250
2017-11-27 16:25:00.000,Martinique,14.641528129577637,-61.024173736572266,7,243,250
2017-11-27 16:25:00.000,Egypt,26.820552825927738,30.80249786376953,1,249,250
2017-11-27 16:25:00.000,Nepal,28.39485740661621,84.12400817871094,4,246,250
2017-11-27 16:25:00.000,Togo,8.619543075561523,0.8247820138931273,3,247,250
2017-11-27 16:25:00.000,Portugal,39.399871826171875,-8.224453926086426,1,249,250
2017-11-27 16:25:00.000,Chile,-35.675148010253906,-71.54296875,0,250,250
2017-11-27 16:25:00.000,Bahrain,25.9304141998291,50.63777160644531,4,246,250
2017-11-27 16:25:00.000,Madagascar,-18.76694679260254,46.86910629272461,0,250,250
2017-11-27 16:25:00.000,Greece,39.0742073059082,21.82431221008301,1,249,250
2017-11-27 16:25:00.000,Mauritius,-20.348403930664066,57.55215072631836,10,240,250
2017-11-27 16:25:00.000,Montserrat,16.74249839782715,-62.1873664855957,6,244,250
2017-11-27 16:25:00.000,Mozambique,-18.66569519042969,35.529563903808594,0,250,250
2017-11-27 16:25:00.000,Norway,60.472023010253906,8.46894645690918,0,250,250
2017-11-27 16:25:00.000,Venezuela,6.423749923706056,-66.58972930908203,0,250,250
2017-11-27 16:25:00.000,Saint Kitts and ,17.35782241821289,-62.78299713134766,6,244,250
```

This is what the analyzed result looks like in S3



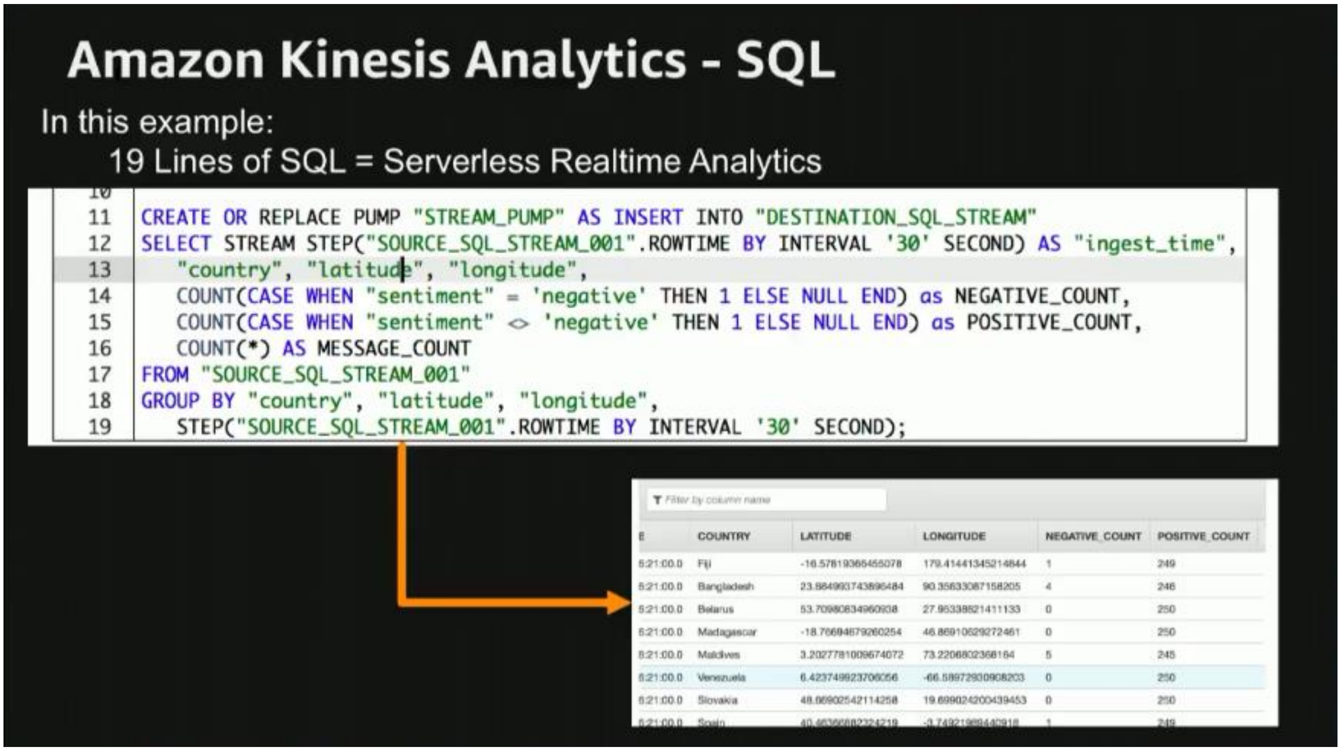
Lambda Pre-processing

```
{"channel":"default","name":"terry.carroll","message":"I discovered the military hospital","timestamp":1511715437343,"country":"Fiji","latitude":-16.57819366455078,"longitude":179.41441345214844}

,{"channel":"default","name":"price.huels","message":"I plans to cease fire against a national chin","timestamp":1511713336760,"country":"Croatia","latitude":45.099998474121094,"longitude":15.199999809265137}
```

```
{"sentiment_value": 0.0, "name": "terry.carroll", "sentiment": "neutral", "timestamp": 1511715437343, "longitude": 179.41441345214844, "country": "Fiji", "latitude": -16.57819366455078, "message": "I discovered the military hospital", "channel": "default"}
,{"sentiment_value": -0.34, "name": "price.huels", "sentiment": "negative", "timestamp": 1511713336760, "longitude": 15.199999809265137, "country": "Croatia", "latitude": 45.099998474121094, "message": "I plans to cease fire against a national chin", "channel": "default"}
,{"sentiment_value": 0.0, "name": "ezekiel.bayer", "sentiment": "neutral", "timestamp": 1511712744775, "longitude": 73.2206802368164, "country": "Maldives", "latitude": 3.2027781009674072, "message": "Last evening i
```

sentiment_value REAL	name VARCHAR(32)	sentiment VARCHAR(8)	COL_timestamp BIGINT	lor DC
0.0	terry.carroll	neutral	1511715437343	17
-0.34	price.huels	negative	1511713336760	15.
0.0	ezekiel.bayer	neutral	1511712744775	73.
-0.2732	ena.rohan	negative	1511714538162	15.
0.0	willis.swaniawski	neutral	1511713856878	9.5

Transform, Enrich, Filter



Amazon Kinesis Analytics - SQL

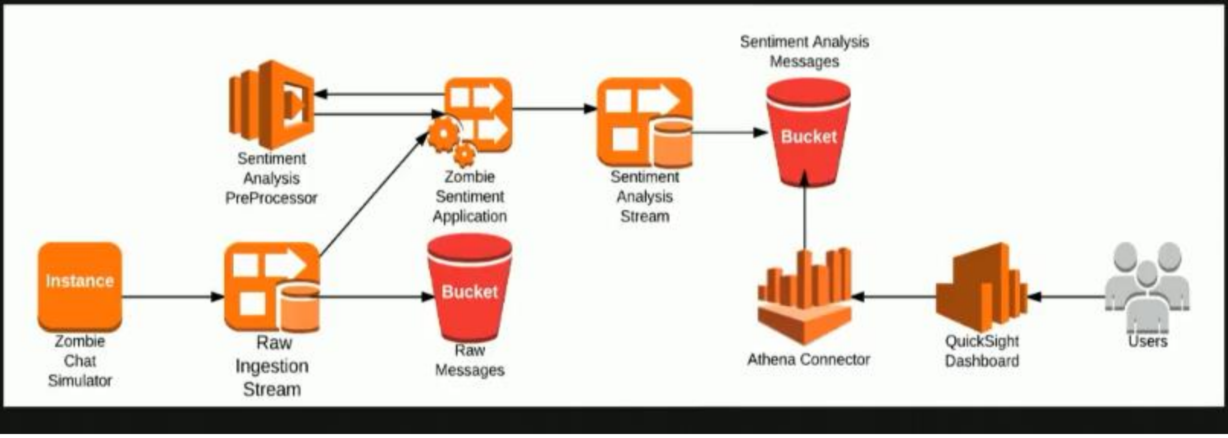
In this example:

19 Lines of SQL = Serverless Realtime Analytics

```
CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT STREAM STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '30' SECOND) AS "ingest_time",
   "country", "latitude", "longitude",
   COUNT(CASE WHEN "sentiment" = 'negative' THEN 1 ELSE NULL END) as NEGATIVE_COUNT,
   COUNT(CASE WHEN "sentiment" <> 'negative' THEN 1 ELSE NULL END) as POSITIVE_COUNT,
   COUNT(*) AS MESSAGE_COUNT
FROM "SOURCE_SQL_STREAM_001"
GROUP BY "country", "latitude", "longitude",
   STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '30' SECOND);
```

E	COUNTRY	LATITUDE	LONGITUDE	NEGATIVE_COUNT	POSITIVE_COUNT
5:21:00.0	Fiji	-16.57819366455078	179.41441345214844	1	249
5:21:00.0	Bangladesh	23.884993743896484	90.35633087158205	4	246
5:21:00.0	Belarus	53.70980834960938	27.95338821411133	0	250
5:21:00.0	Madagascar	-18.76694679260254	46.86910629272461	0	250
5:21:00.0	Maldives	3.2027781009674072	73.2206802368164	5	245
5:21:00.0	Venezuela	6.423749923706056	-66.58972930908203	0	250
5:21:00.0	Slovakia	48.66902542114258	19.699024200439453	0	250
5:21:00.0	Spain	40.46366882324219	-3.7492198944091 8	1	249

Larger Picture of what we showed:



Sentiment Analysis Messages

Sentiment Analysis PreProcessor

Zombie Sentiment Application

Sentiment Analysis Stream

Bucket

Instance

Zombie Chat Simulator

Raw Ingestion Stream

Bucket

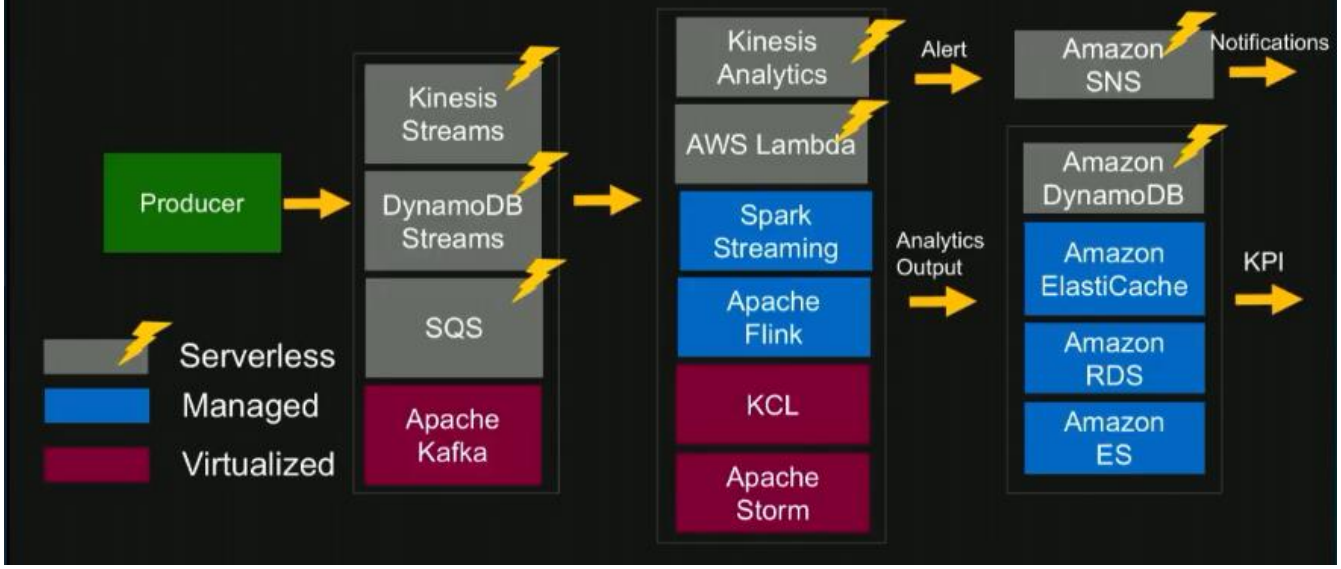
Raw Messages

Athena Connector

QuickSight Dashboard

Users

Fitting into existing Real-time Analytics



Producer

Kinesis Streams

DynamoDB Streams

SQS

Apache Kafka

Kinesis Analytics

AWS Lambda

Spark Streaming

Apache Flink

KCL

Apache Storm

Alert

Analytics Output

Amazon SNS

Notifications

Amazon DynamoDB

Amazon ElastiCache

Amazon RDS

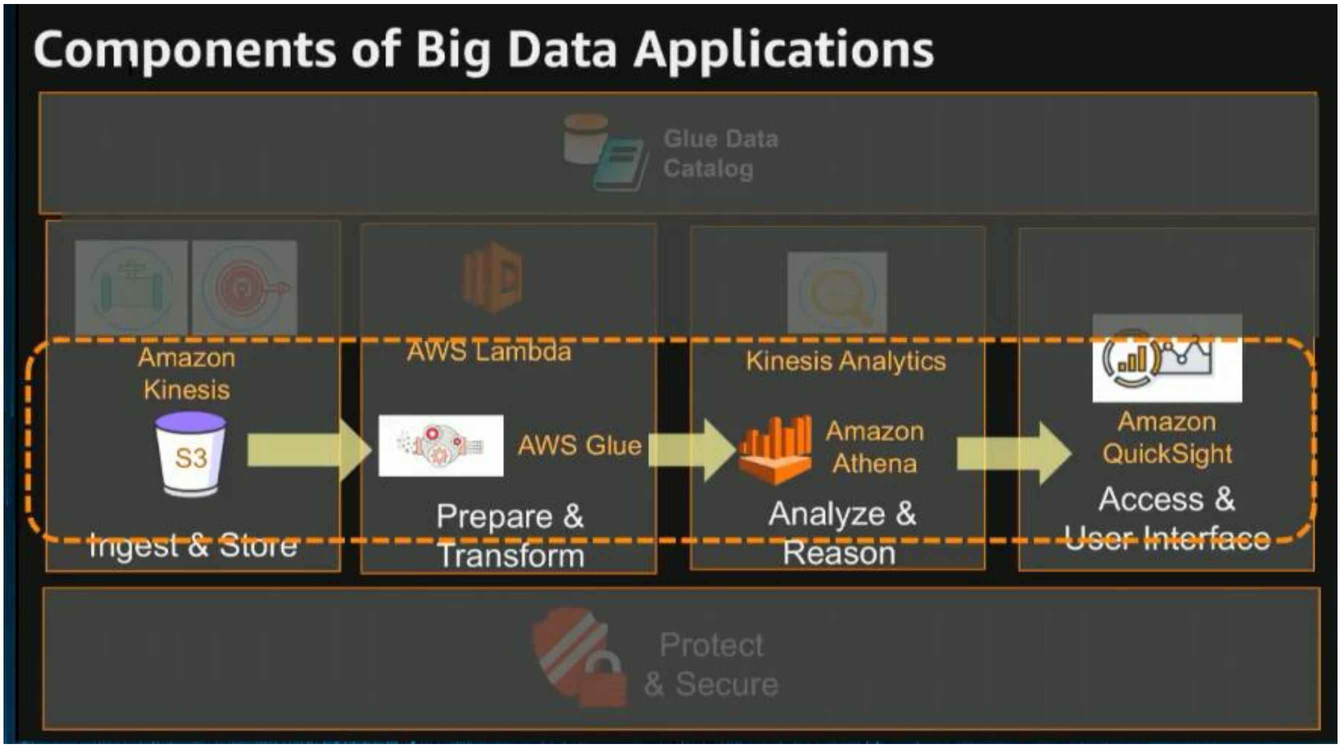
Amazon ES

KPI

Serverless

Managed

Virtualized



Components of Big Data Applications

Glue Data Catalog

Amazon Kinesis

S3

Ingest & Store

AWS Lambda

AWS Glue

Prepare & Transform

Kinesis Analytics

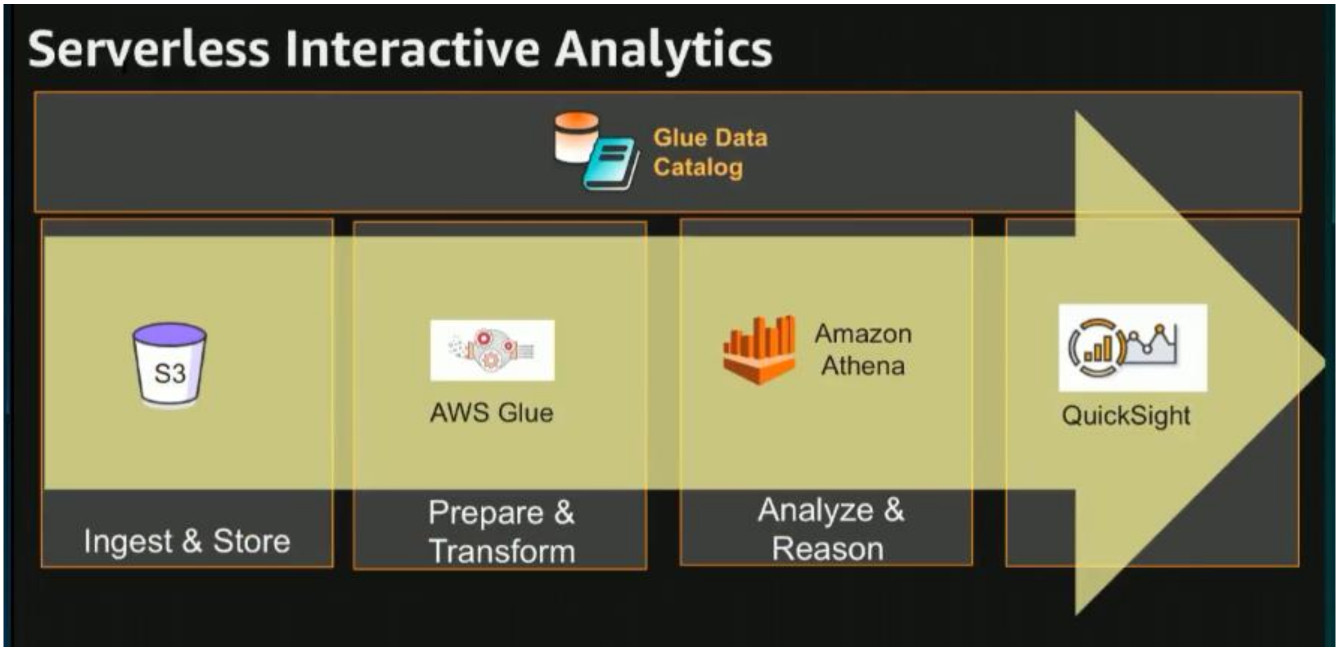
Amazon Athena

Analyze & Reason

Amazon QuickSight

Access & User Interface

Protect & Secure



Serverless Interactive Analytics

Glue Data Catalog

S3

Ingest & Store

AWS Glue

Prepare & Transform

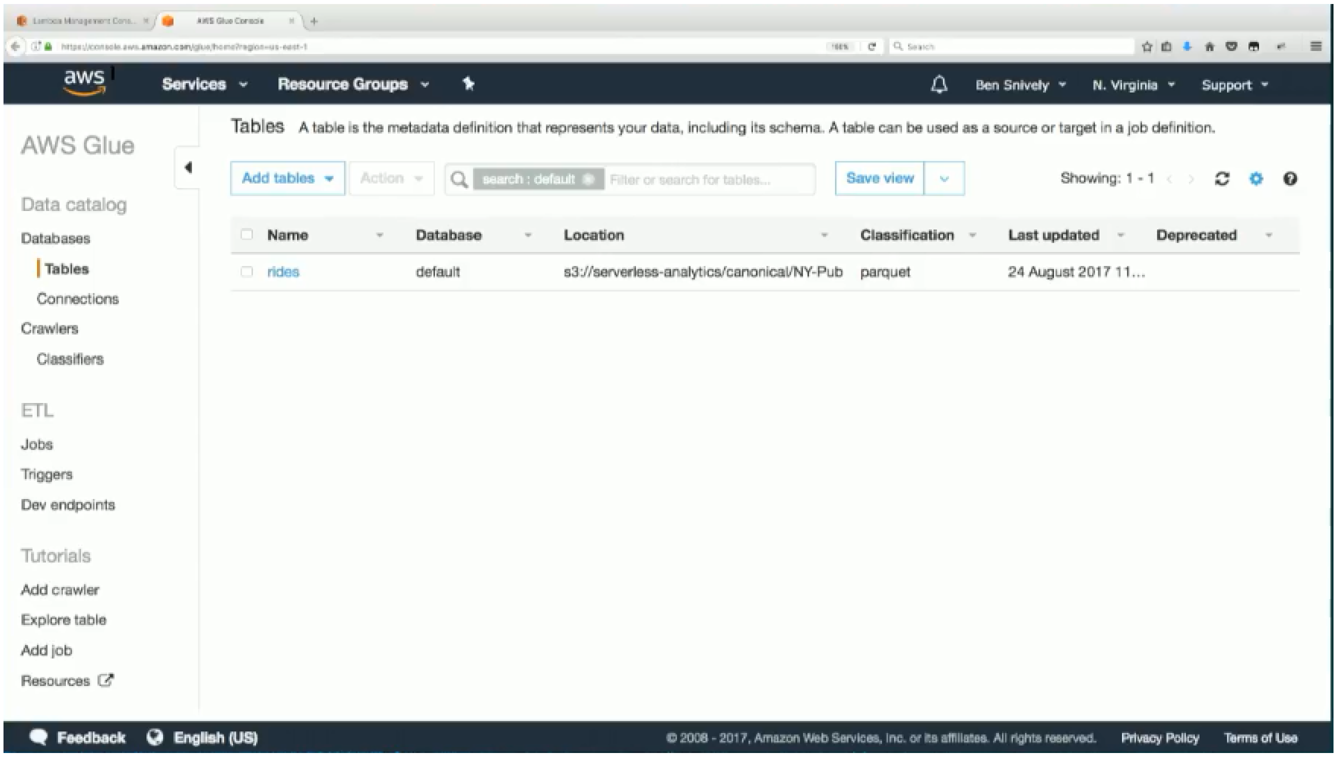
Amazon Athena

Analyze & Reason

QuickSight

Demonstration

Let us now do some interactive analytics on some of our data at rest in S3.

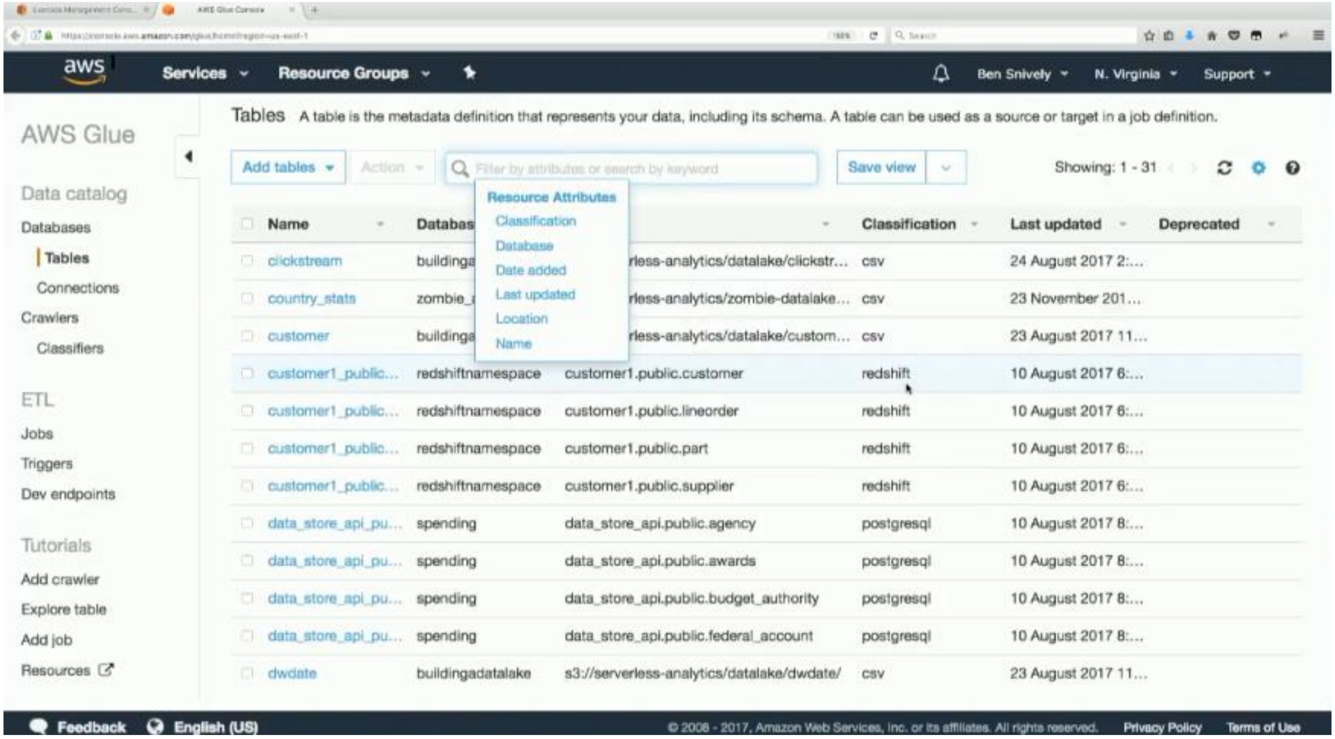


Tables A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

Name	Database	Location	Classification	Last updated	Deprecated
rides	default	s3://serverless-analytics/canonical/NY-Pub	parquet	24 August 2017 11...	

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

We are starting with the Glue catalog to help describe our data on S3

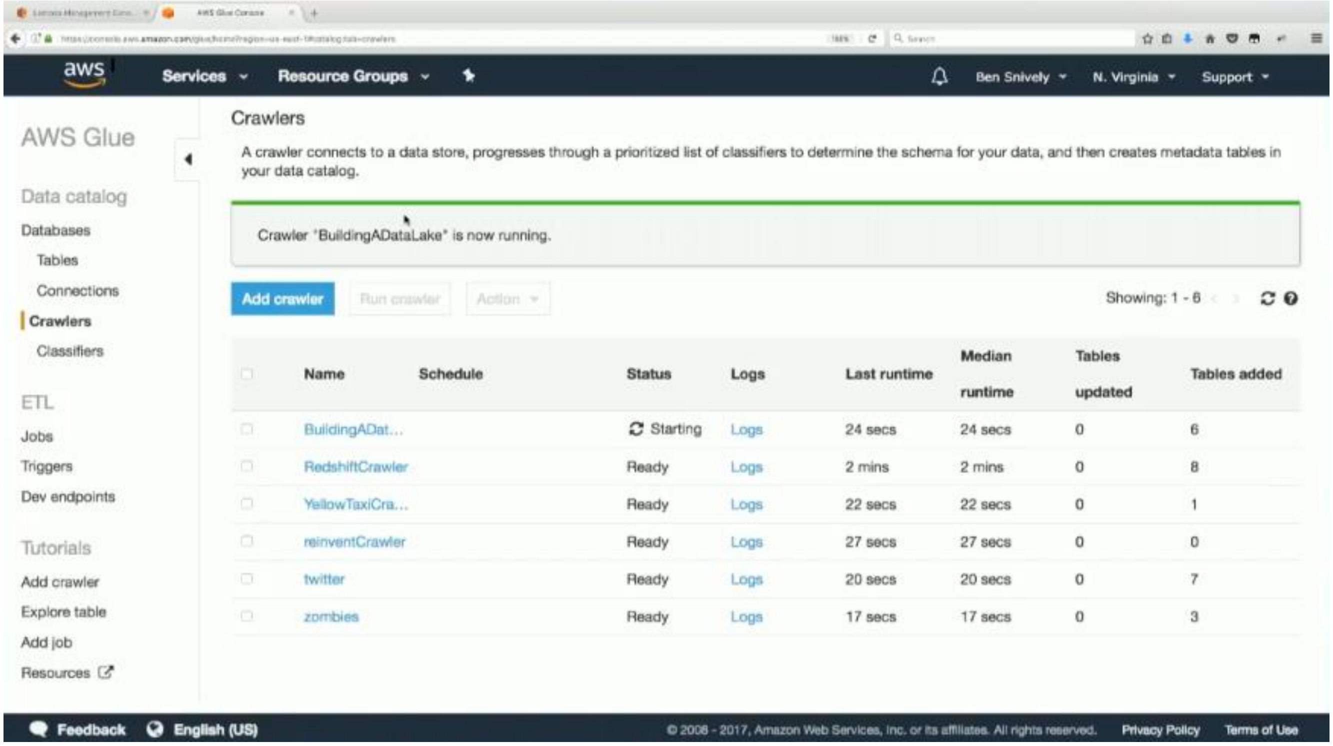


Tables A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

Name	Databas		Classification	Last updated	Deprecated
clickstream	buildinga	rless-analytics/datalake/clickstr...	csv	24 August 2017 2:...	
country_stats	zombie_	rless-analytics/zombie-datalake...	csv	23 November 201...	
customer	buildinga	rless-analytics/datalake/custom...	csv	23 August 2017 11...	
customer1_public...	redshiftnamespace	customer1.public.customer	redshift	10 August 2017 6:...	
customer1_public...	redshiftnamespace	customer1.public.lineorder	redshift	10 August 2017 6:...	
customer1_public...	redshiftnamespace	customer1.public.part	redshift	10 August 2017 6:...	
customer1_public...	redshiftnamespace	customer1.public.supplier	redshift	10 August 2017 6:...	
data_store_api_pu...	spending	data_store_api.public.agency	postgresql	10 August 2017 8:...	
data_store_api_pu...	spending	data_store_api.public.awards	postgresql	10 August 2017 8:...	
data_store_api_pu...	spending	data_store_api.public.budget_authority	postgresql	10 August 2017 8:...	
data_store_api_pu...	spending	data_store_api.public.federal_account	postgresql	10 August 2017 8:...	
dwdate	buildingadatalake	s3://serverless-analytics/datalake/dwdate/	csv	23 August 2017 11...	

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

There are actually several data type available in the S3 location, we can filter the data by type



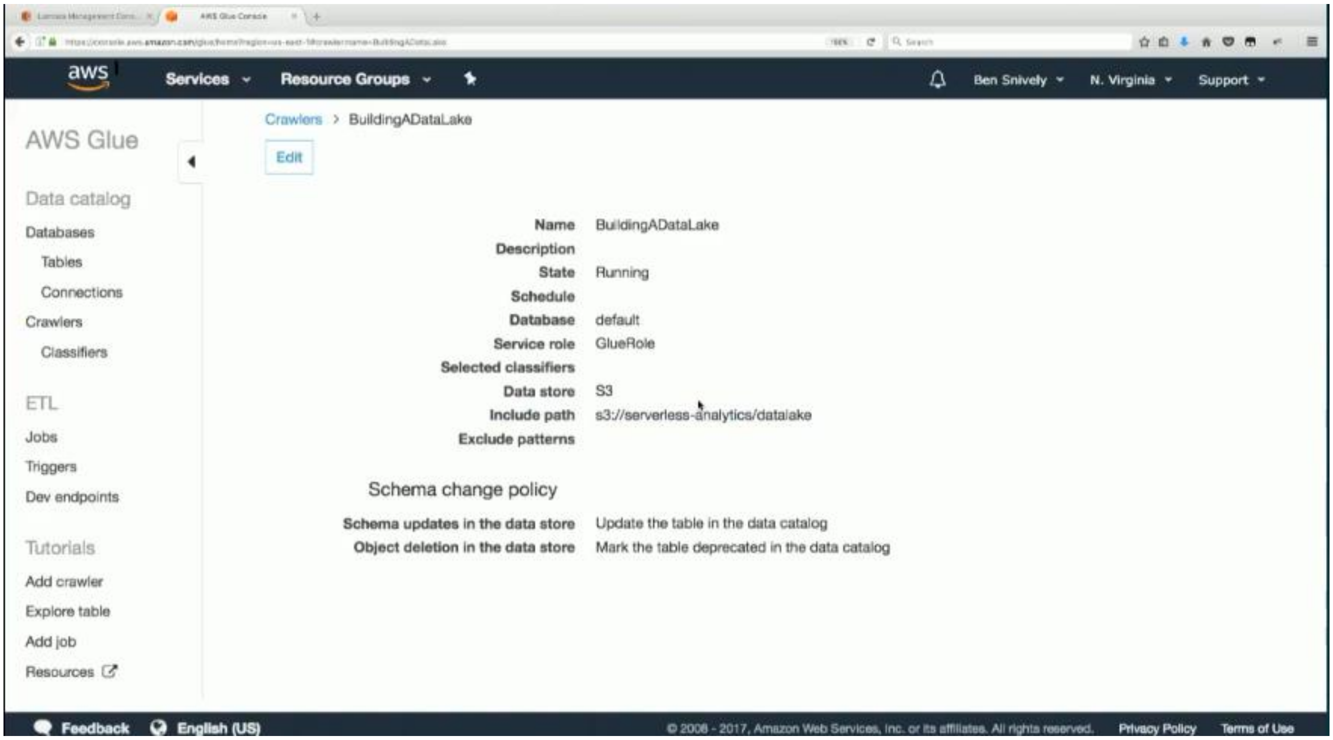
Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawler "BuildingADataLake" is now running.

Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
BuildingADat...		Starting	Logs	24 secs	24 secs	0	6
RedshiftCrawler		Ready	Logs	2 mins	2 mins	0	8
YellowTaxiCra...		Ready	Logs	22 secs	22 secs	0	1
reinventCrawler		Ready	Logs	27 secs	27 secs	0	0
twitter		Ready	Logs	20 secs	20 secs	0	7
zombies		Ready	Logs	17 secs	17 secs	0	3

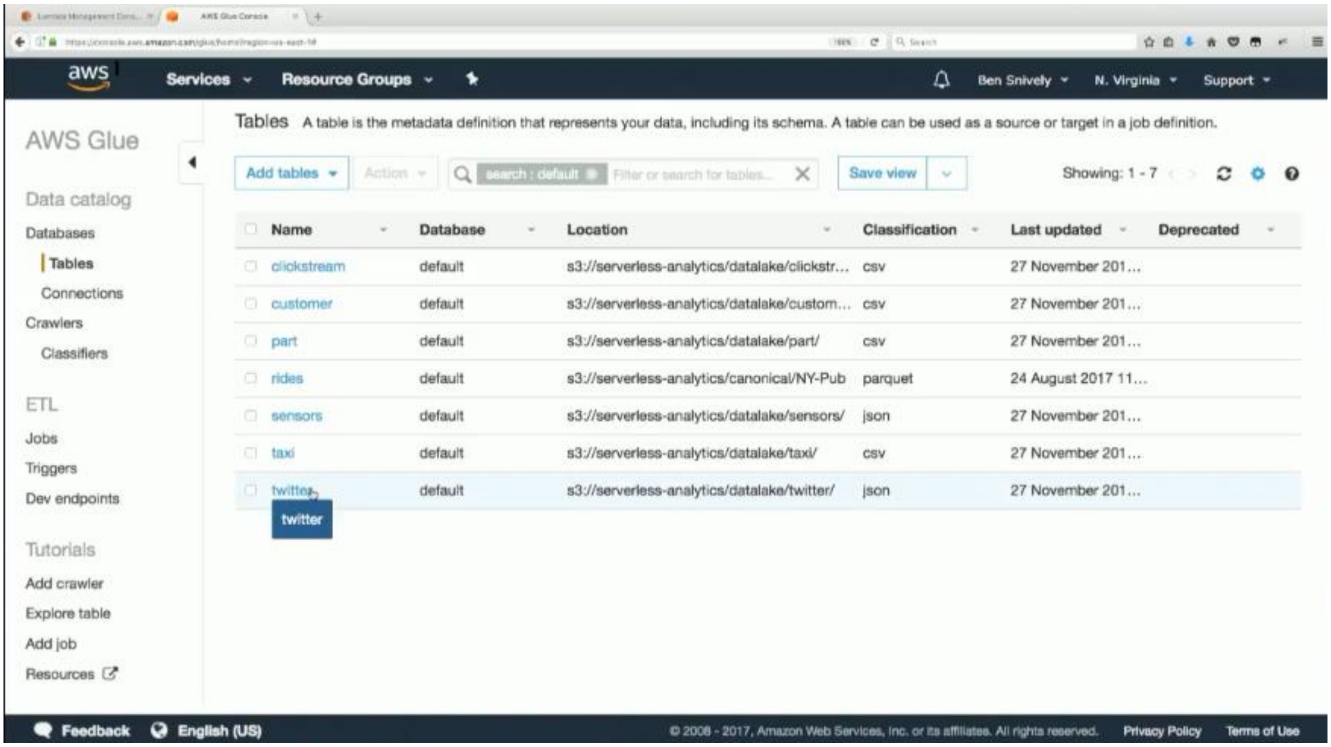
We can use Glue to crawl a new S3 location, the crawlers run with an IAM role that we set up for the crawler to allow it crawl our data on S3 to discover all the new tables in there



Name	BuildingADataLake
Description	
State	Running
Schedule	
Database	default
Service role	GlueRole
Selected classifiers	
Data store	S3
Include path	s3://serverless-analytics/datalake
Exclude patterns	

Schema change policy

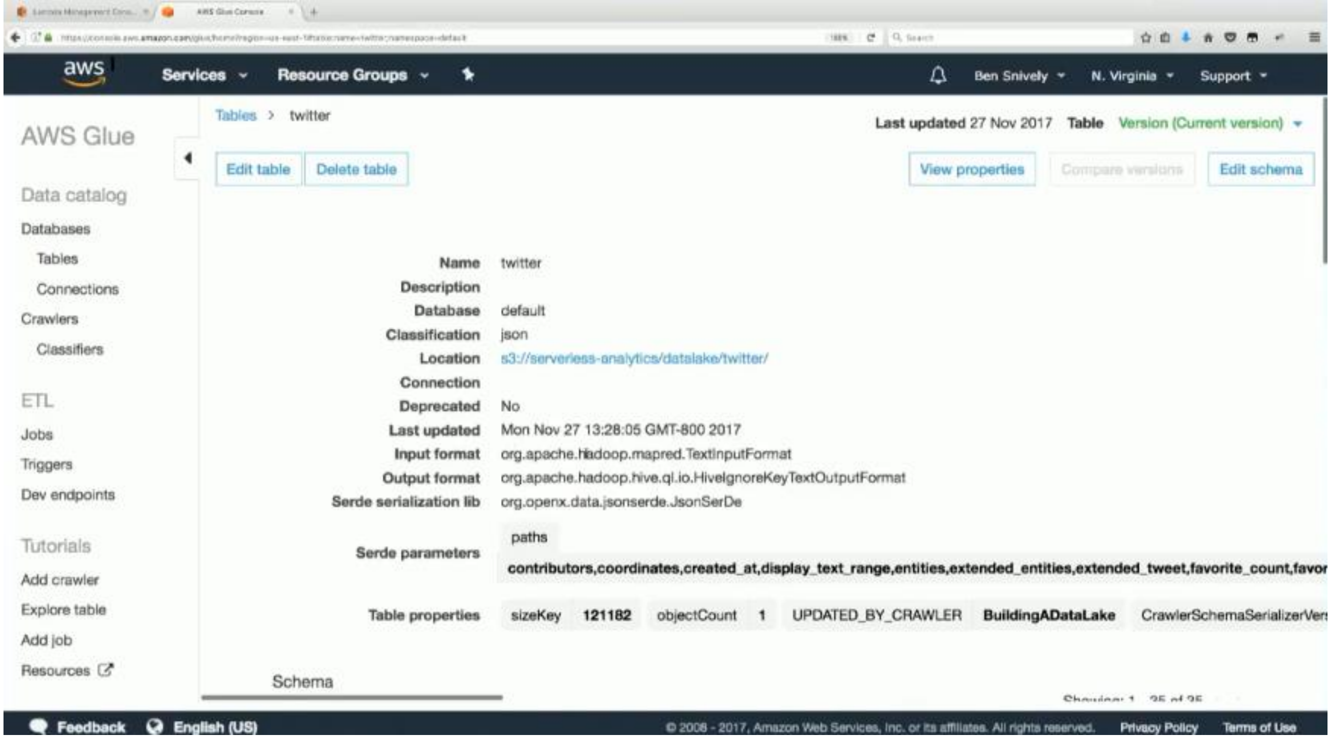
Schema updates in the data store	Update the table in the data catalog
Object deletion in the data store	Mark the table deprecated in the data catalog



Tables A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

Name	Database	Location	Classification	Last updated	Deprecated
clickstream	default	s3://serverless-analytics/datalake/clickstr…	csv	27 November 201…	
customer	default	s3://serverless-analytics/datalake/custom…	csv	27 November 201…	
part	default	s3://serverless-analytics/datalake/part/	csv	27 November 201…	
rides	default	s3://serverless-analytics/canonical/NY-Pub	parquet	24 August 2017 11…	
sensors	default	s3://serverless-analytics/datalake/sensors/	json	27 November 201…	
taxi	default	s3://serverless-analytics/datalake/taxi/	csv	27 November 201…	
twitter	default	s3://serverless-analytics/datalake/twitter/	json	27 November 201…	

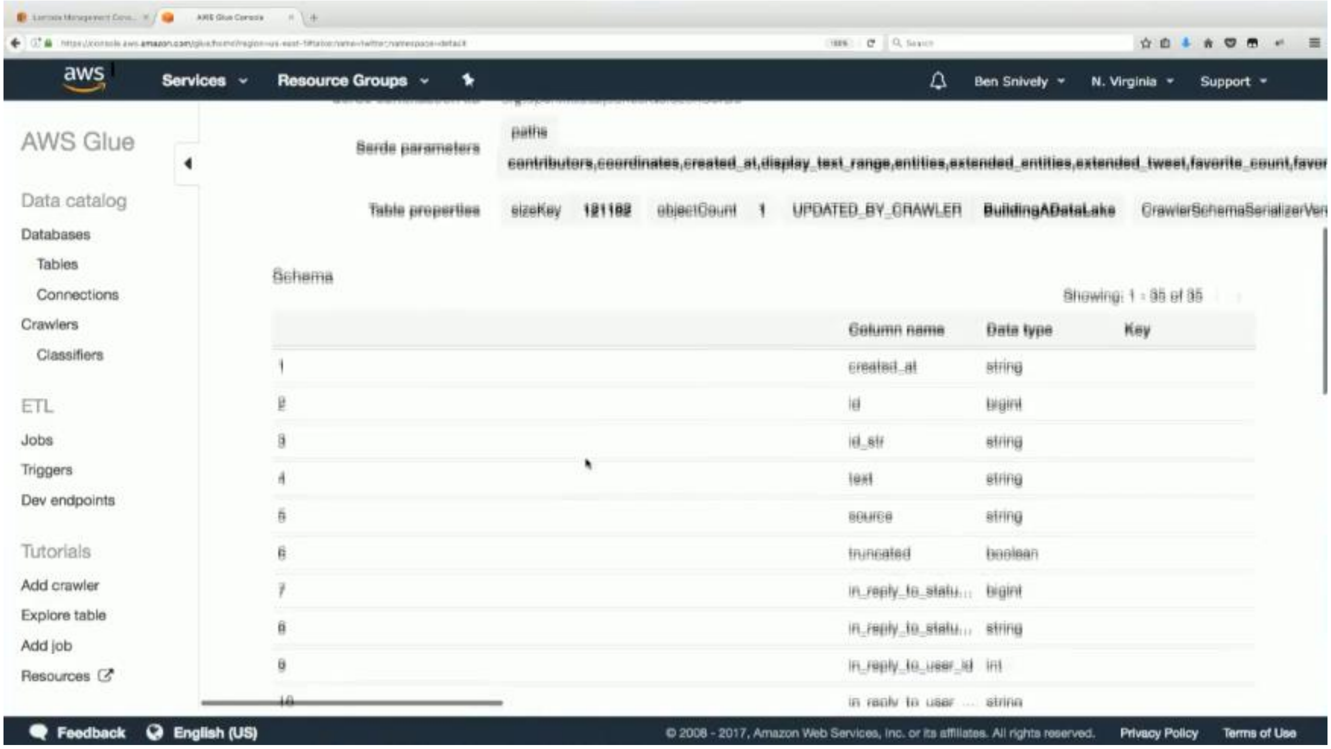
We now have several tables created for us by the crawler with the data sets discovered



Last updated 27 Nov 2017 Table Version (Current version)

Name	twitter
Description	
Database	default
Classification	json
Location	s3://serverless-analytics/datalake/twitter/
Connection	
Deprecated	No
Last updated	Mon Nov 27 13:28:05 GMT-800 2017
Input format	org.apache.hadoop.mapred.TextInputFormat
Output format	org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Serde serialization lib	org.openx.data.jsonserde.JsonSerDe
Serde parameters	paths contributors,coordinates,created_at,display_text_range,entities,extended_entities,extended_tweet,favorite_count,favor
Table properties	sizeKey 121182 objectCount 1 UPDATED_BY_CRAWLER BuildingADataLake CrawlerSchemaSerializerVer

This is the twitter data it discovered, we can now use this in our analytics ad queries



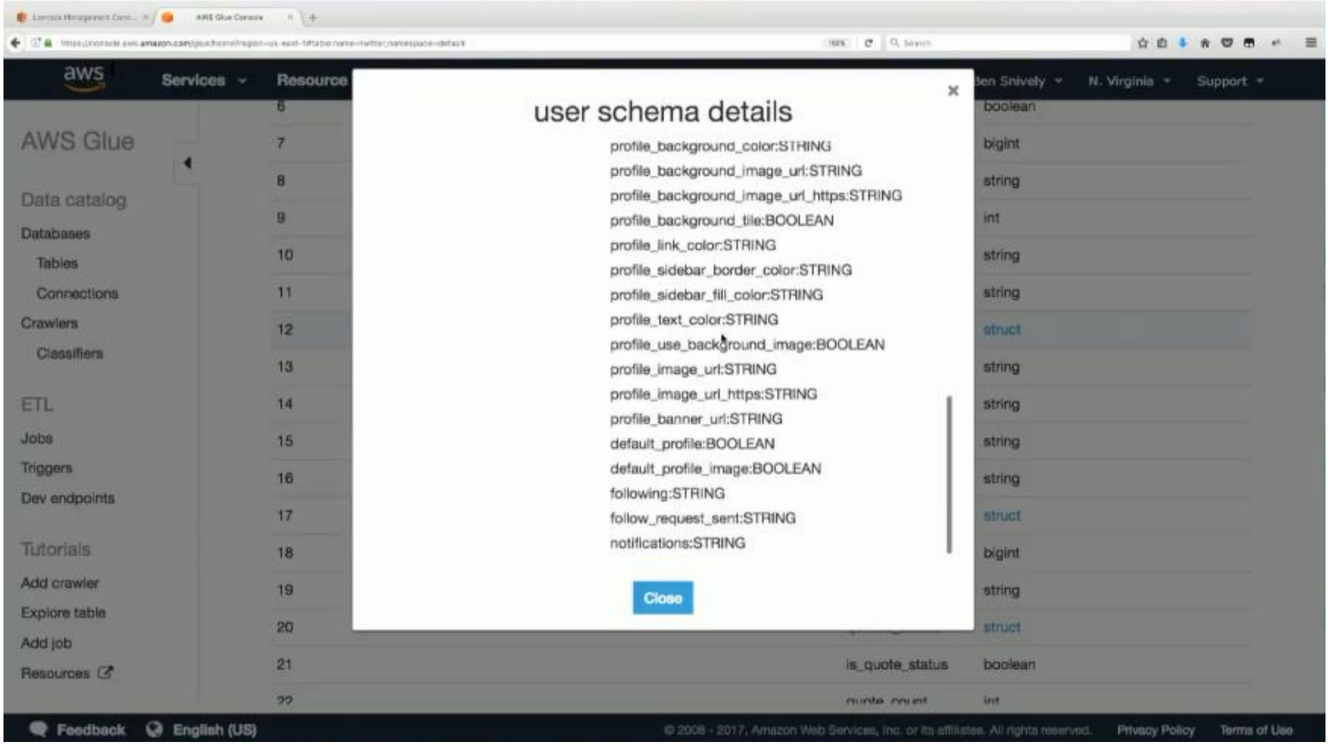
Serde parameters: paths contributors,coordinates,created_at,display_text_range,entities,extended_entities,extended_tweet,favorite_count,favor

Table properties: sizeKey 121182 objectCount 1 UPDATED_BY_CRAWLER BuildingADataLake CrawlerSchemaSerializerVer

Schema

Showing: 1 - 35 of 35

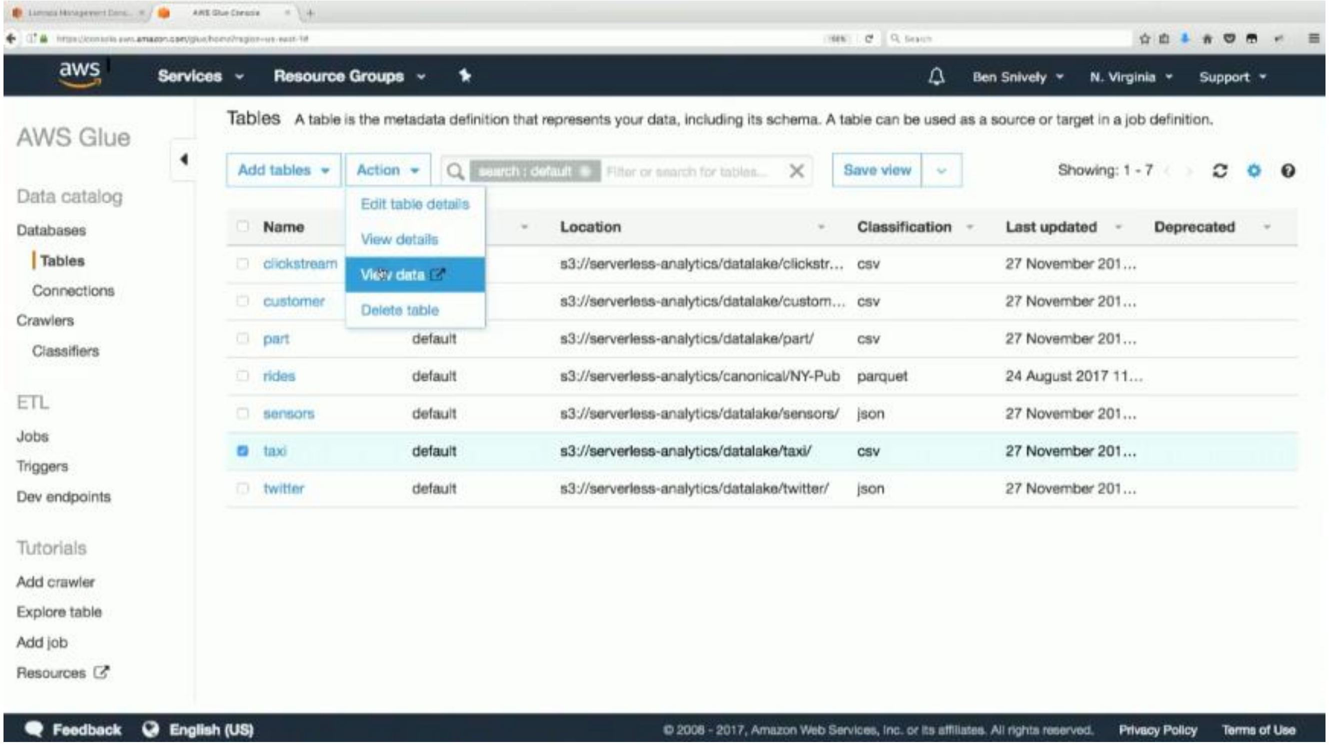
	Column name	Data type	Key
1	created_at	string	
2	id	bigint	
3	id_str	string	
4	text	string	
5	source	string	
6	truncated	boolean	
7	in_reply_to_statu...	bigint	
8	in_reply_to_statu...	string	
9	in_reply_to_user_id	int	
10	in_reply_to_user ...	string	



user schema details

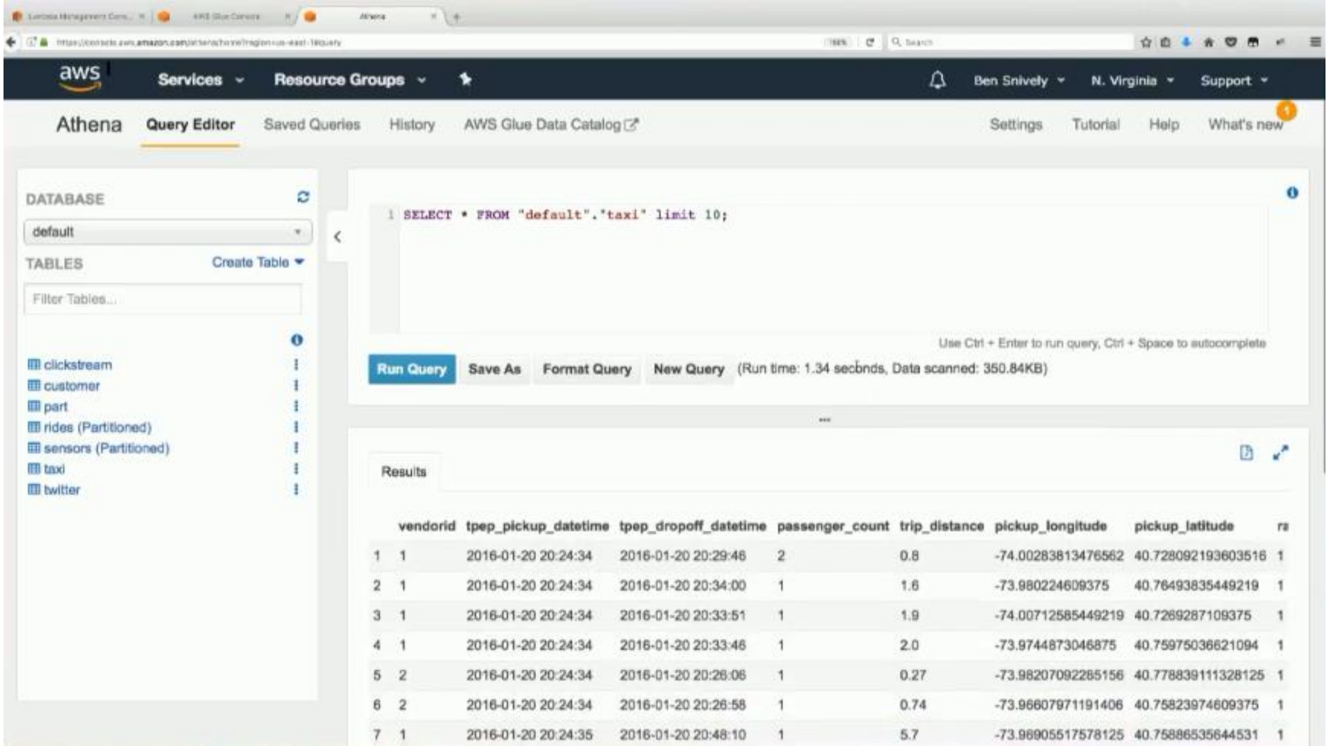
```
profile_background_color:STRING
profile_background_image_url:STRING
profile_background_image_url_https:STRING
profile_background_tile:BOOLEAN
profile_link_color:STRING
profile_sidebar_border_color:STRING
profile_sidebar_fill_color:STRING
profile_text_color:STRING
profile_use_background_image:BOOLEAN
profile_image_url:STRING
profile_image_url_https:STRING
profile_banner_url:STRING
default_profile:BOOLEAN
default_profile_image:BOOLEAN
following:STRING
follow_request_sent:STRING
notifications:STRING
```

It also discovered the JSON that can represent the twitter data, we can modify this schema as we like



Tables A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

	Name		Location	Classification	Last updated	Deprecated
□	clickstream		s3://serverless-analytics/datalake/clickstr...	csv	27 November 201...	
□	customer		s3://serverless-analytics/datalake/custom...	csv	27 November 201...	
□	part	default	s3://serverless-analytics/datalake/part/	csv	27 November 201...	
□	rides	default	s3://serverless-analytics/canonical/NY-Pub	parquet	24 August 2017 11...	
□	sensors	default	s3://serverless-analytics/datalake/sensors/	json	27 November 201...	
☑	taxi	default	s3://serverless-analytics/datalake/taxi/	csv	27 November 201...	
□	twitter	default	s3://serverless-analytics/datalake/twitter/	json	27 November 201...	



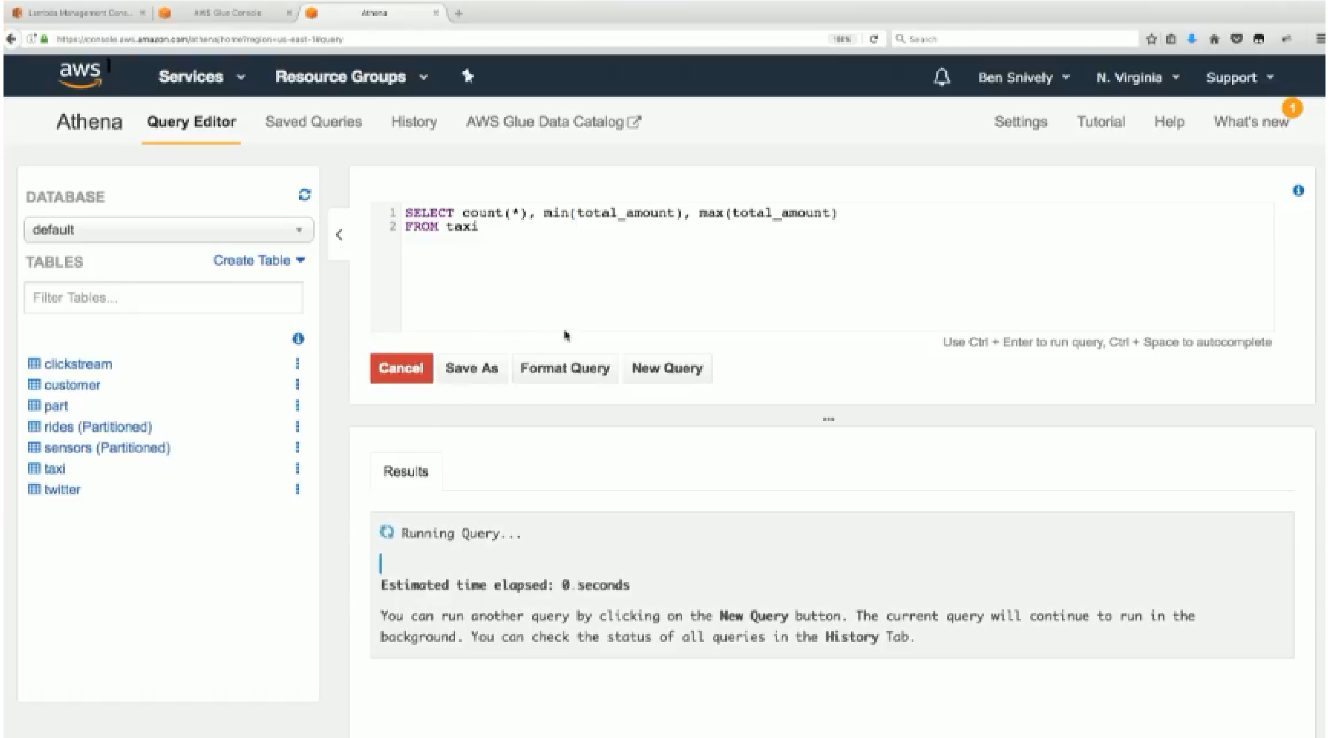
```
SELECT * FROM "default"."taxi" limit 10;
```

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

(Run time: 1.34 seconds, Data scanned: 350.84KB)

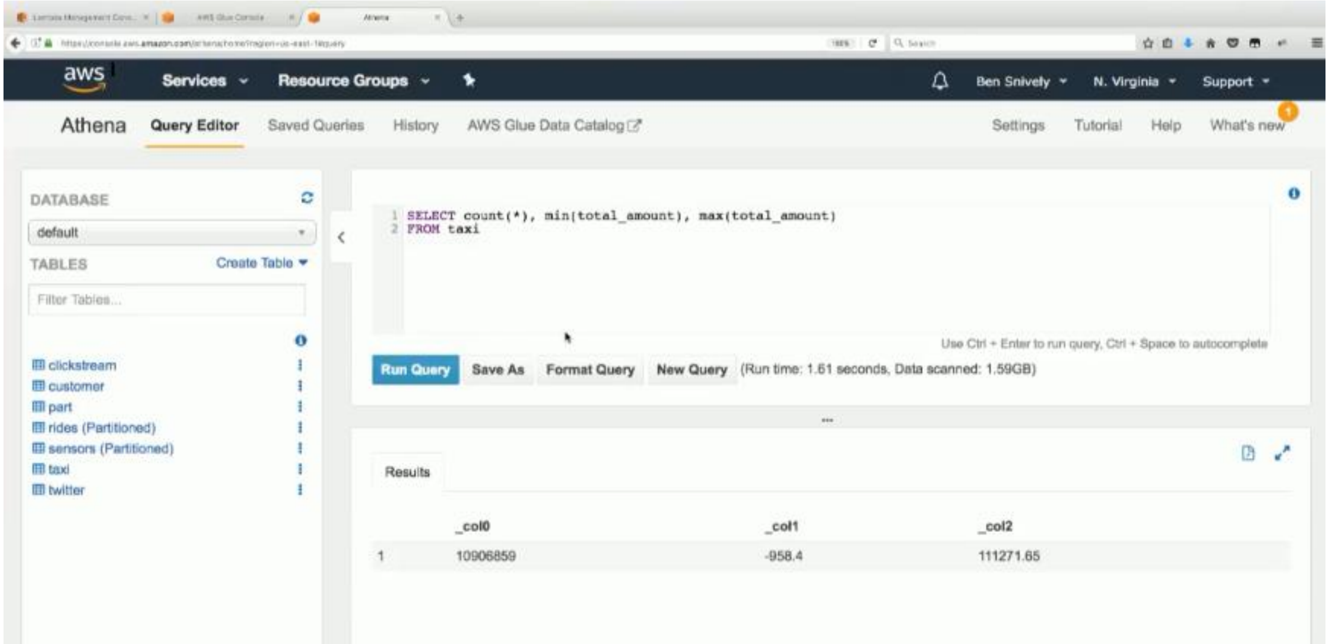
	vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_latitude	ra
1	1	2016-01-20 20:24:34	2016-01-20 20:29:46	2	0.8	-74.00283813476562	40.728092193603516	1
2	1	2016-01-20 20:24:34	2016-01-20 20:34:00	1	1.6	-73.980224609375	40.76493835449219	1
3	1	2016-01-20 20:24:34	2016-01-20 20:33:51	1	1.9	-74.00712585449219	40.7269287109375	1
4	1	2016-01-20 20:24:34	2016-01-20 20:33:46	1	2.0	-73.9744873046875	40.75975036621094	1
5	2	2016-01-20 20:24:34	2016-01-20 20:26:06	1	0.27	-73.98207092285156	40.778839111328125	1
6	2	2016-01-20 20:24:34	2016-01-20 20:26:58	1	0.74	-73.96607971191406	40.75823974609375	1
7	1	2016-01-20 20:24:35	2016-01-20 20:48:10	1	5.7	-73.96905517578125	40.75886535644531	1

Now we can use Athena to start querying the data using SQL to analyze this dataset using SELECT commands.



```
SELECT count(*), min(total_amount), max(total_amount)
FROM taxi
```

```
Running Query...

Estimated time elapsed: 0 seconds

You can run another query by clicking on the New Query button. The current query will continue to run in the
background. You can check the status of all queries in the History Tab.
```

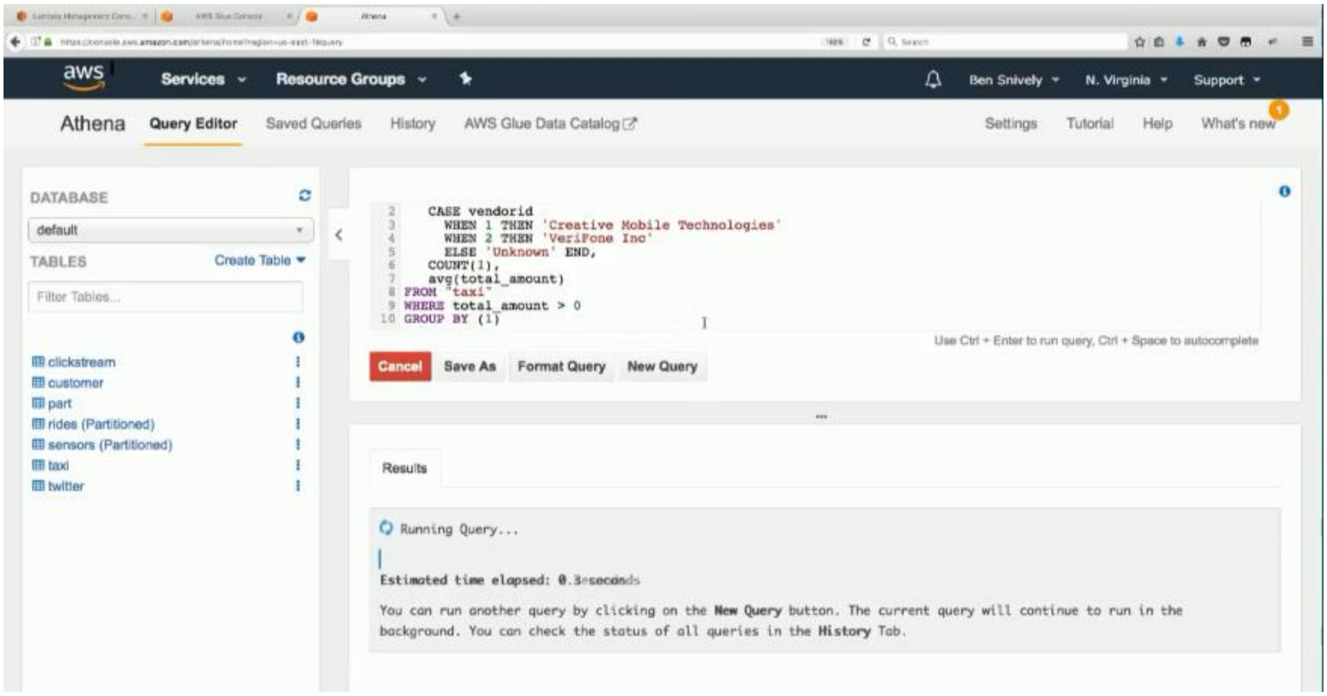


```
SELECT count(*), min(total_amount), max(total_amount)
FROM taxi
```

(Run time: 1.61 seconds, Data scanned: 1.59GB)

	_col0	_col1	_col2
1	10906859	-958.4	111271.65

We can see that we have over 10 million records available for this month



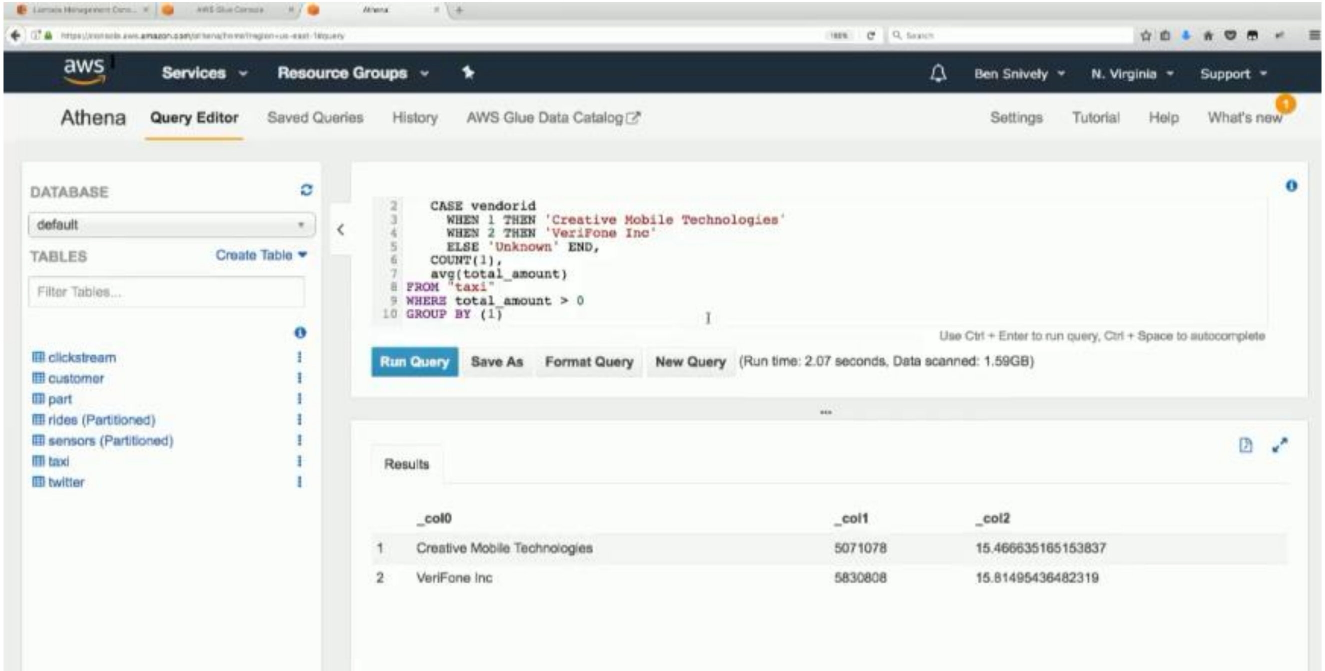
```
  CASE vendorid
    WHEN 1 THEN 'Creative Mobile Technologies'
    WHEN 2 THEN 'VeriFone Inc'
    ELSE 'Unknown' END,
  COUNT(1),
  avg(total_amount)
FROM "taxi"
WHERE total_amount > 0
GROUP BY (1)
```

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

Running Query...

Estimated time elapsed: 0.3 seconds

You can run another query by clicking on the **New Query** button. The current query will continue to run in the background. You can check the status of all queries in the **History** Tab.



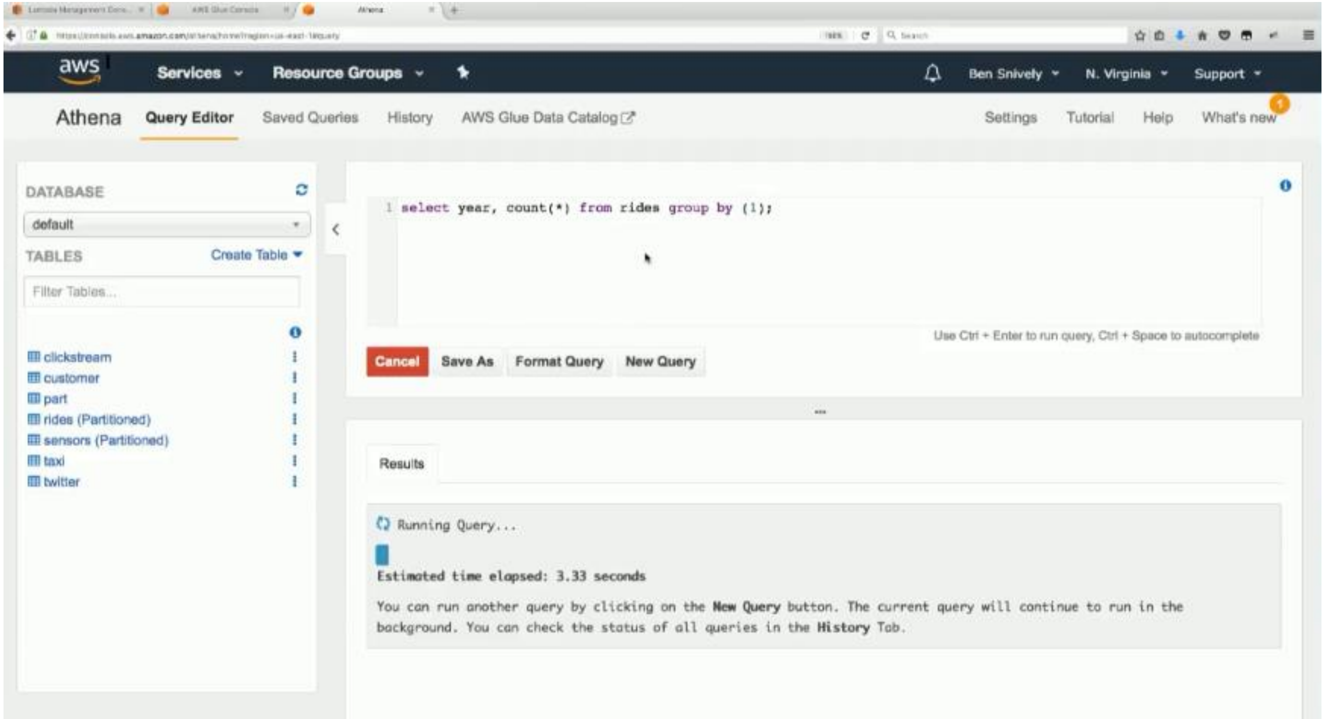
```
  CASE vendorid
    WHEN 1 THEN 'Creative Mobile Technologies'
    WHEN 2 THEN 'VeriFone Inc'
    ELSE 'Unknown' END,
  COUNT(1),
  avg(total_amount)
FROM "taxi"
WHERE total_amount > 0
GROUP BY (1)
```

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

(Run time: 2.07 seconds, Data scanned: 1.59GB)

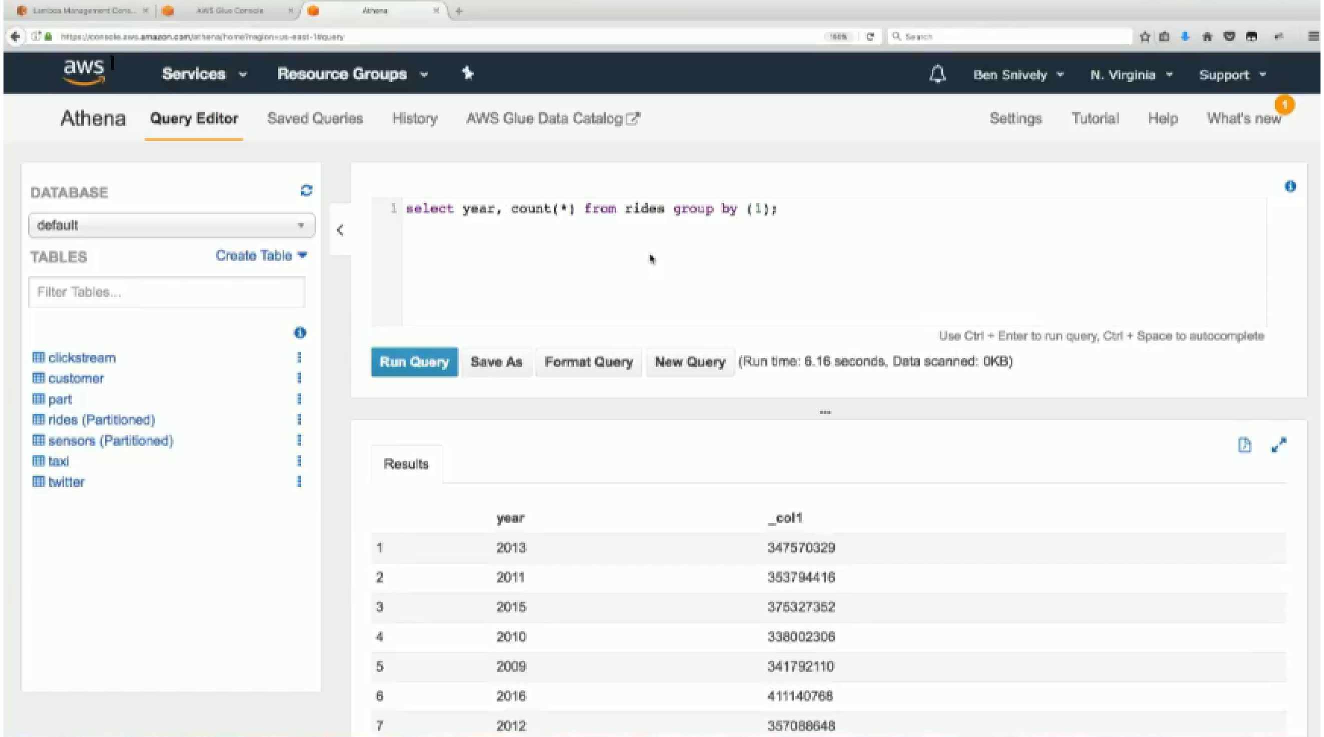
	_col0	_col1	_col2
1	Creative Mobile Technologies	5071078	15.466635165153837
2	VeriFone Inc	5830808	15.81495436482319

The results come back very fast



```
select year, count(*) from rides group by (1);
```

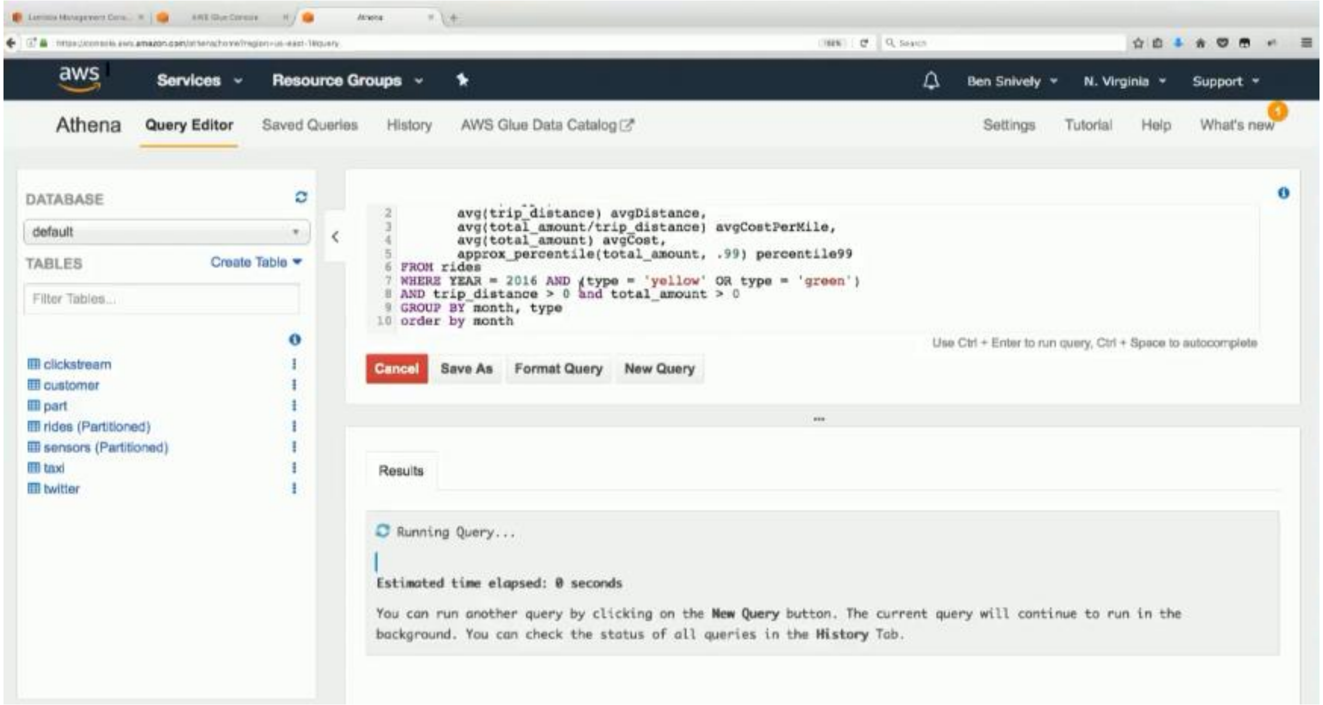
```
Running Query...

Estimated time elapsed: 3.33 seconds

You can run another query by clicking on the New Query button. The current query will continue to run in the
background. You can check the status of all queries in the History Tab.
```



```
select year, count(*) from rides group by (1);
```

(Run time: 6.16 seconds, Data scanned: 0KB)

	year	_col1
1	2013	347570329
2	2011	353794416
3	2015	375327352
4	2010	338002306
5	2009	341792110
6	2016	411140768
7	2012	357088648

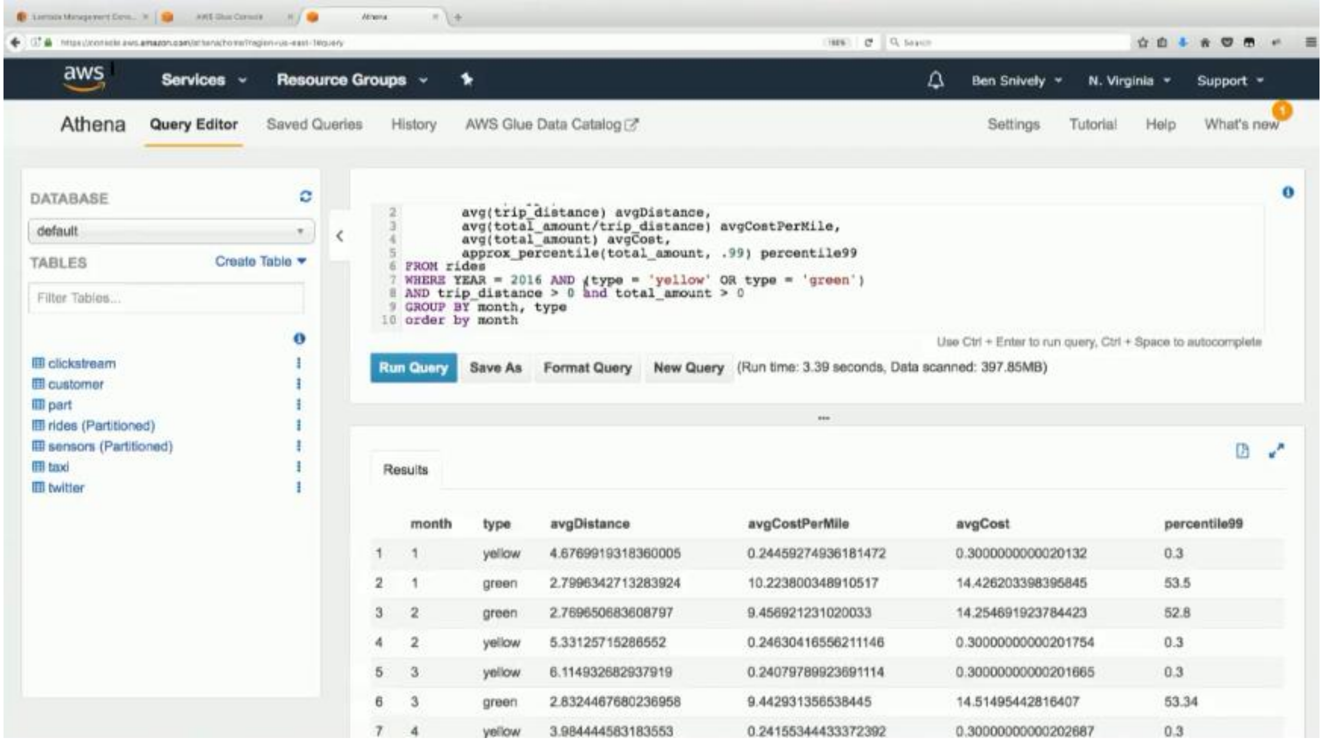


```
        avg(trip_distance) avgDistance,
        avg(total_amount/trip_distance) avgCostPerMile,
        avg(total_amount) avgCost,
        approx_percentile(total_amount, .99) percentile99
FROM rides
WHERE YEAR = 2016 AND (type = 'yellow' OR type = 'green')
AND trip_distance > 0 and total_amount > 0
GROUP BY month, type
order by month
```

Running Query...

Estimated time elapsed: 0 seconds

You can run another query by clicking on the **New Query** button. The current query will continue to run in the background. You can check the status of all queries in the **History** Tab.

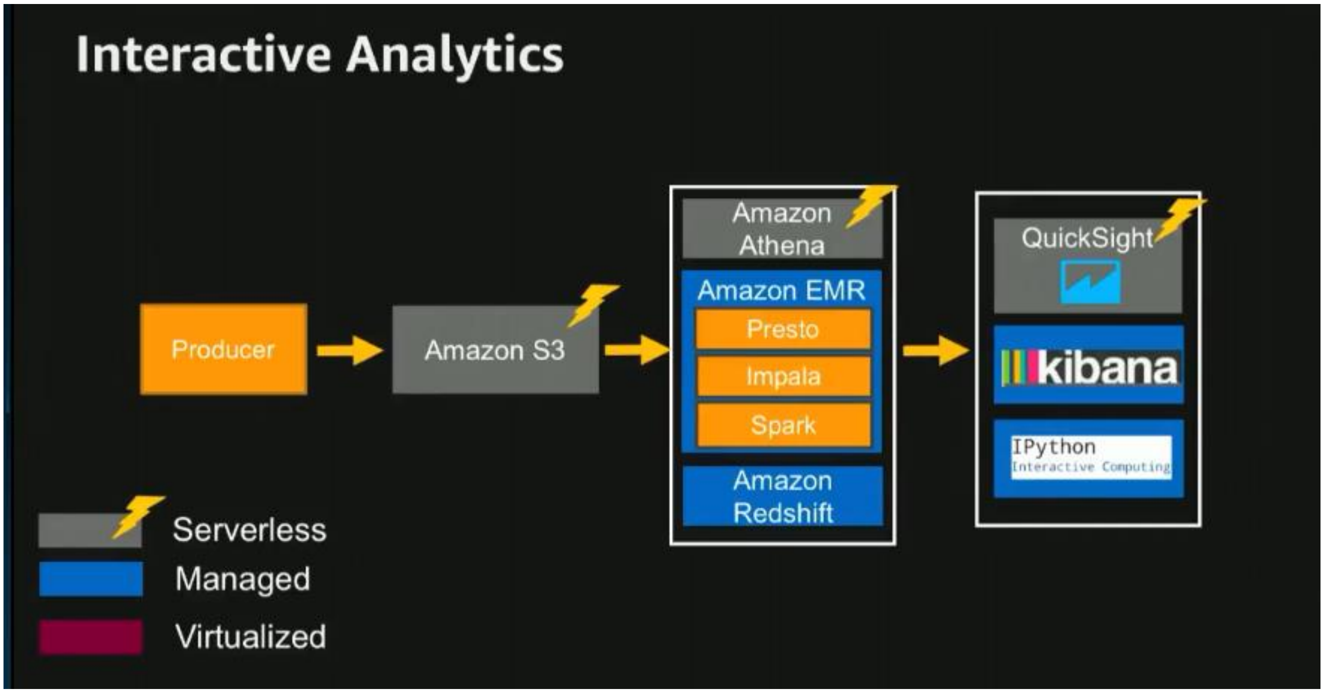


```
        avg(trip_distance) avgDistance,
        avg(total_amount/trip_distance) avgCostPerMile,
        avg(total_amount) avgCost,
        approx_percentile(total_amount, .99) percentile99
FROM rides
WHERE YEAR = 2016 AND (type = 'yellow' OR type = 'green')
AND trip_distance > 0 and total_amount > 0
GROUP BY month, type
order by month
```

(Run time: 3.39 seconds, Data scanned: 397.85MB)

	month	type	avgDistance	avgCostPerMile	avgCost	percentile99
1	1	yellow	4.6769919318360005	0.24459274936181472	0.3000000000020132	0.3
2	1	green	2.7996342713283924	10.223800348910517	14.426203398395845	53.5
3	2	green	2.769650683608797	9.456921231020033	14.254691923784423	52.8
4	2	yellow	5.33125715286552	0.24630416556211146	0.30000000000201754	0.3
5	3	yellow	6.114932682937919	0.24079789923691114	0.30000000000201665	0.3
6	3	green	2.8324467680236958	9.442931356538445	14.51495442816407	53.34
7	4	yellow	3.984444583183553	0.24155344433372392	0.30000000000202687	0.3

Demonstration



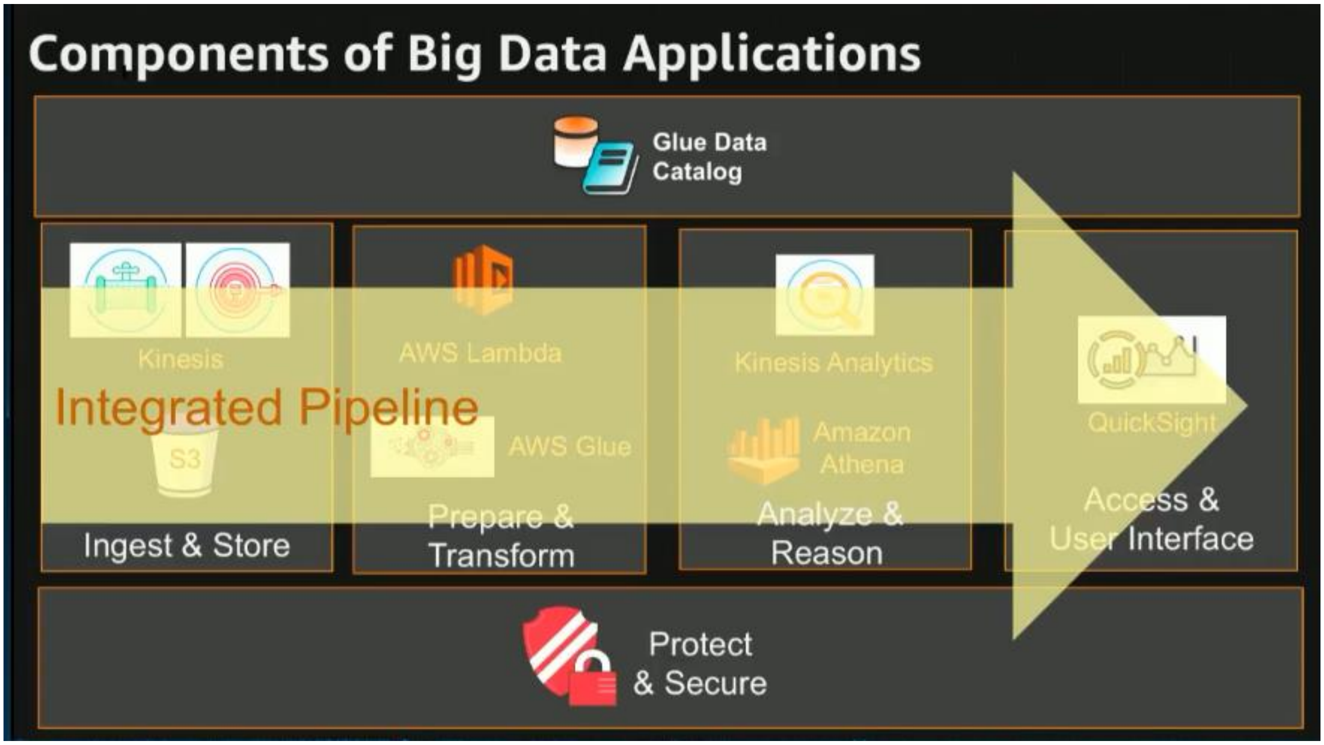
Interactive Analytics

Producer → Amazon S3 → Amazon Athena, Amazon EMR (Presto, Impala, Spark), Amazon Redshift → QuickSight, kibana, IPython Interactive Computing

Serverless

Managed

Virtualized



Components of Big Data Applications

Glue Data Catalog

Kinesis

S3

Ingest & Store

AWS Lambda

AWS Glue

Prepare & Transform

Kinesis Analytics

Amazon Athena

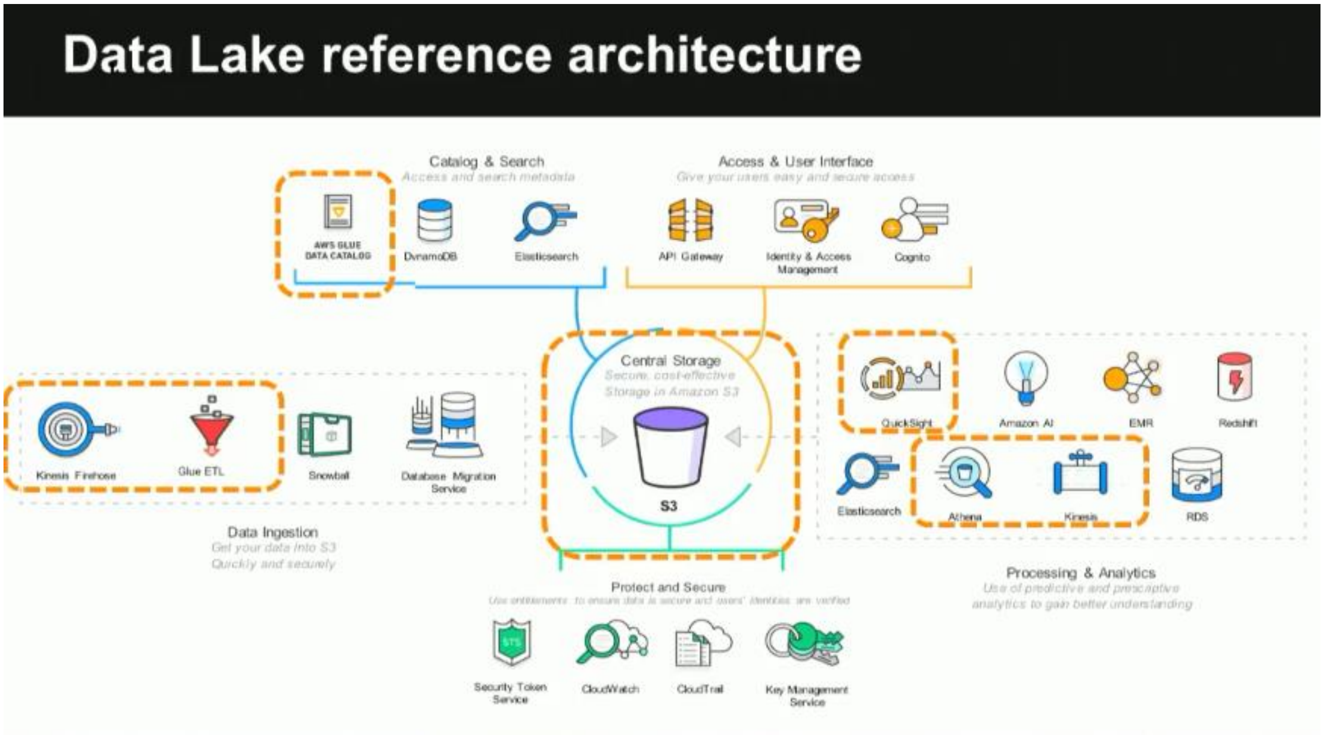
Analyze & Reason

QuickSight

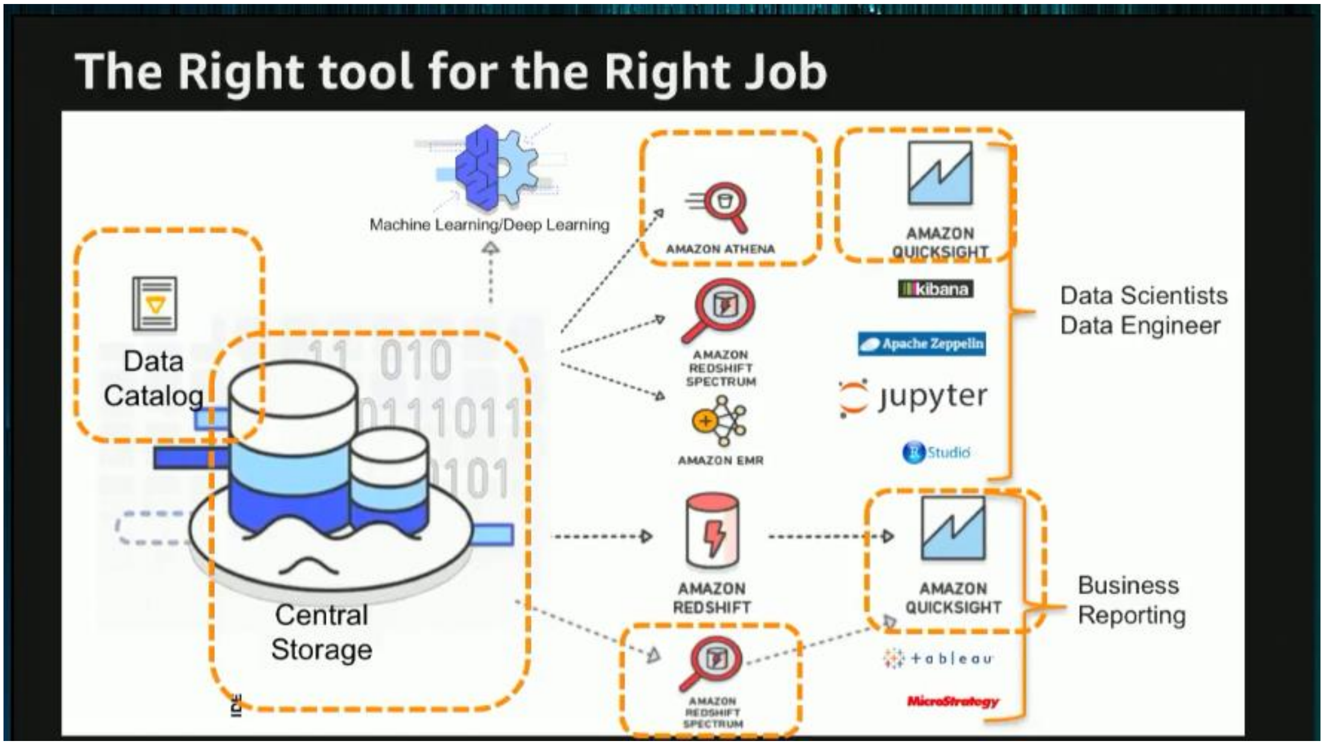
Access & User Interface

Integrated Pipeline

Protect & Secure

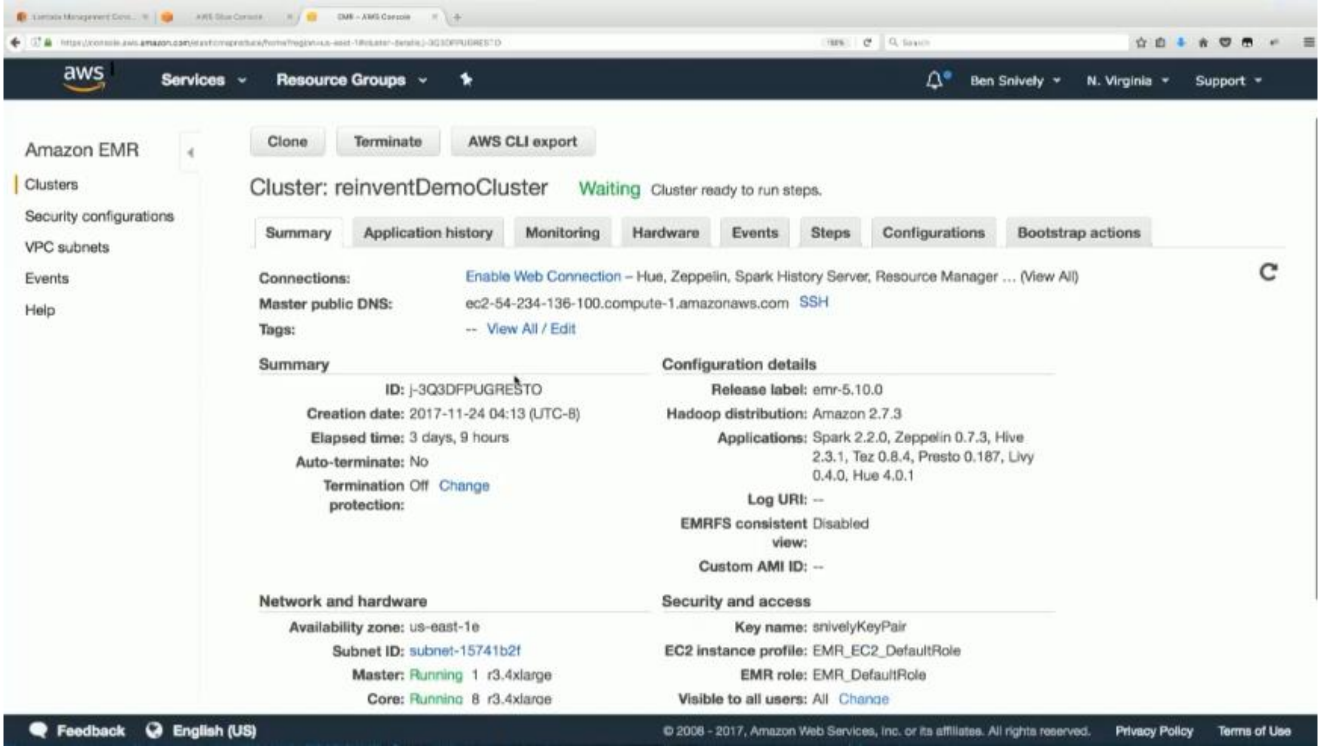


Data Lake reference architecture



The Right tool for the Right Job

What about existing Hadoop Clusters?



Cluster: reinventDemoCluster Waiting Cluster ready to run steps.

Connections: Enable Web Connection – Hue, Zeppelin, Spark History Server, Resource Manager ... (View All)
Master public DNS: ec2-54-234-136-100.compute-1.amazonaws.com SSH
Tags: -- View All / Edit

Summary

ID: j-3Q3DFPUGRESTO
Creation date: 2017-11-24 04:13 (UTC-8)
Elapsed time: 3 days, 9 hours
Auto-terminate: No
Termination protection: Off Change

Configuration details

Release label: emr-5.10.0
Hadoop distribution: Amazon 2.7.3
Applications: Spark 2.2.0, Zeppelin 0.7.3, Hive 2.3.1, Tez 0.8.4, Presto 0.187, Livy 0.4.0, Hue 4.0.1
Log URI: --
EMRFS consistent view: Disabled
Custom AMI ID: --

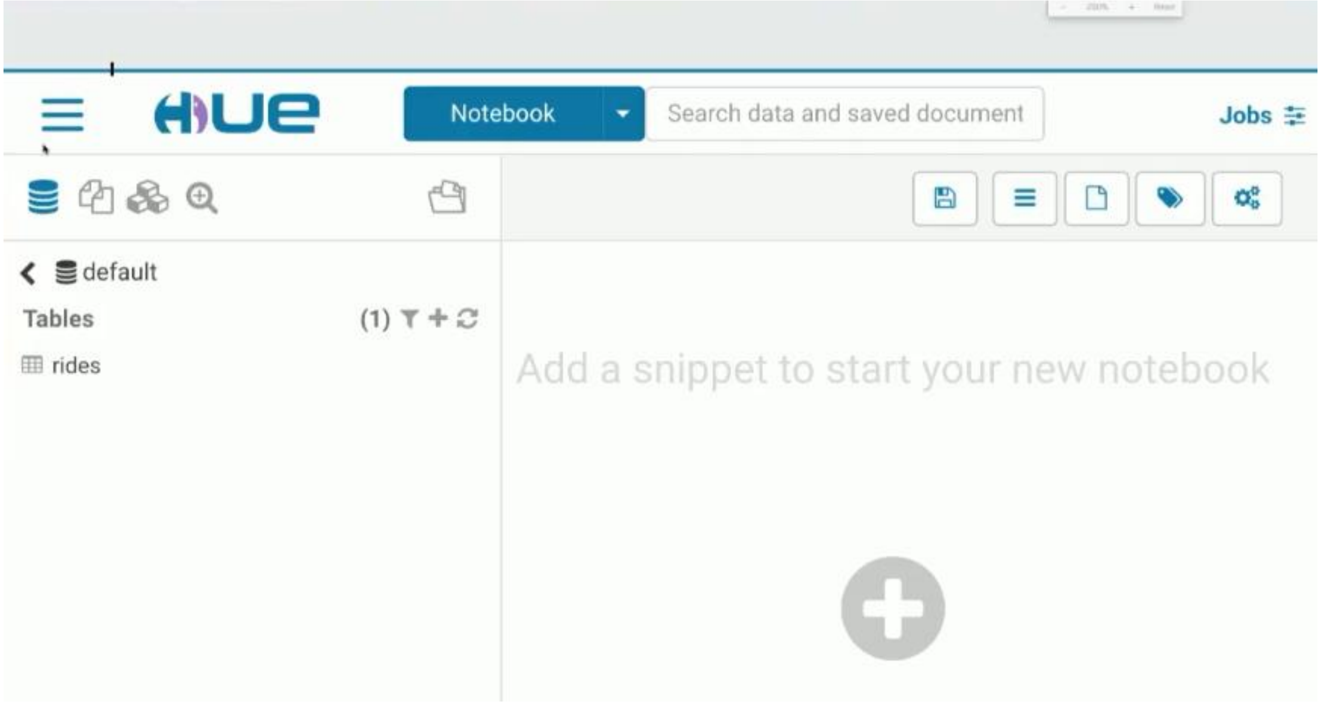
Network and hardware

Availability zone: us-east-1e
Subnet ID: subnet-15741b2f
Master: Running 1 r3.4xlarge
Core: Running 8 r3.4xlarge

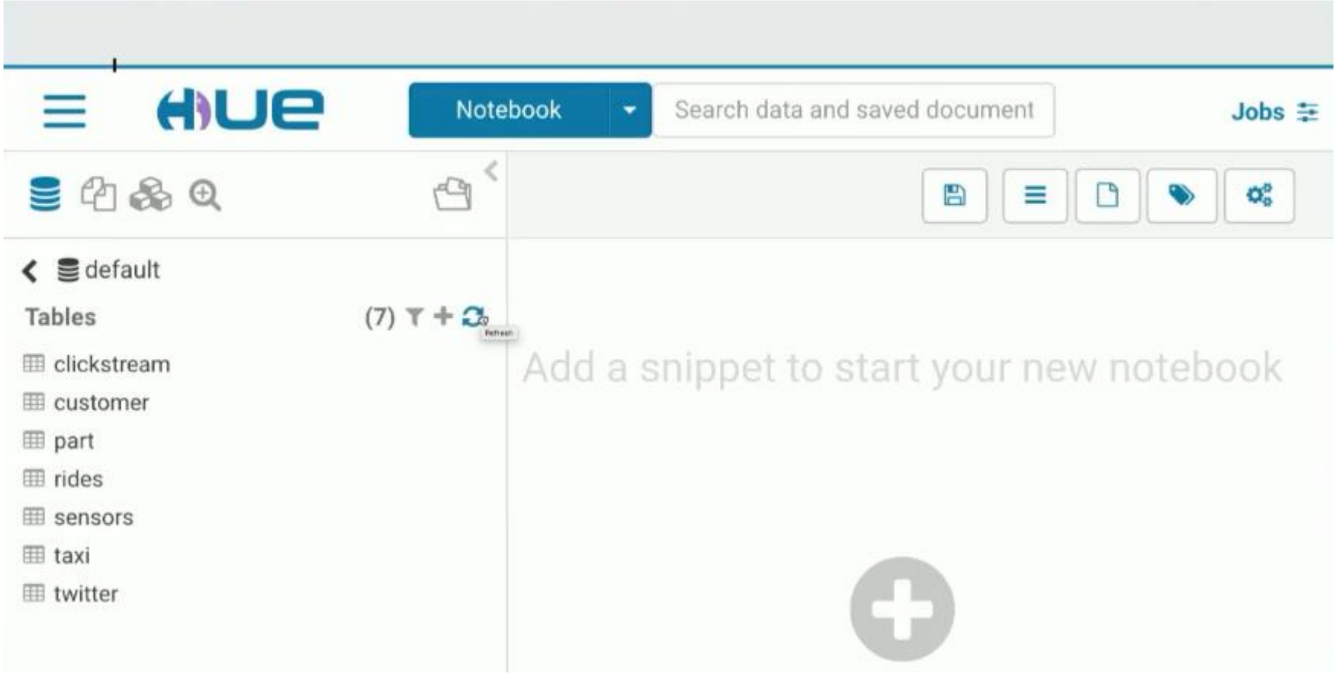
Security and access

Key name: snivelyKeyPair
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All Change

We have our EMR cluster running here, it has several of the ecosystem applications on it, we also specified that it use the Glue data catalog.


HUE
Notebook
Search data and saved document
Jobs
default
Tables
(1)
rides
Add a snippet to start your new notebook

Hue is an interactive web interface for interacting with your Hadoop cluster.



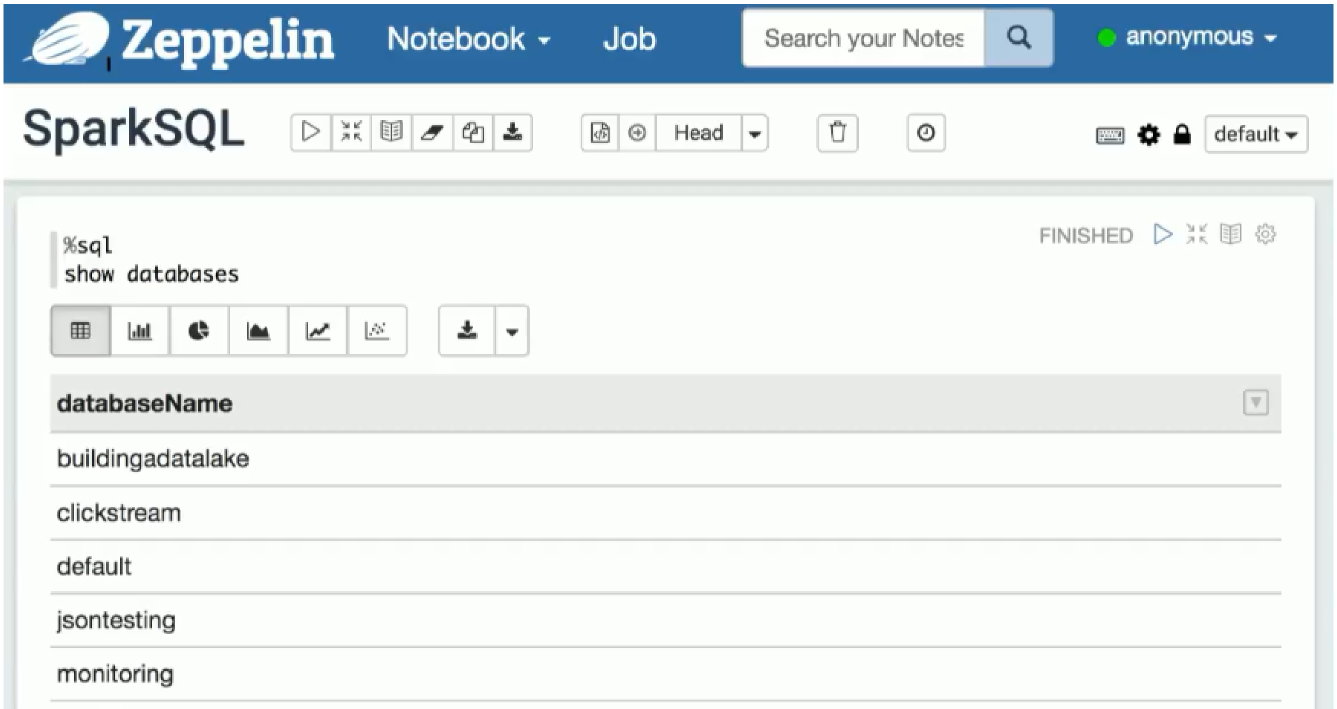
default

Tables (7)

- clickstream
- customer
- part
- rides
- sensors
- taxi
- twitter

Add a snippet to start your new notebook

These are the data tables we discovered with the Glue crawler

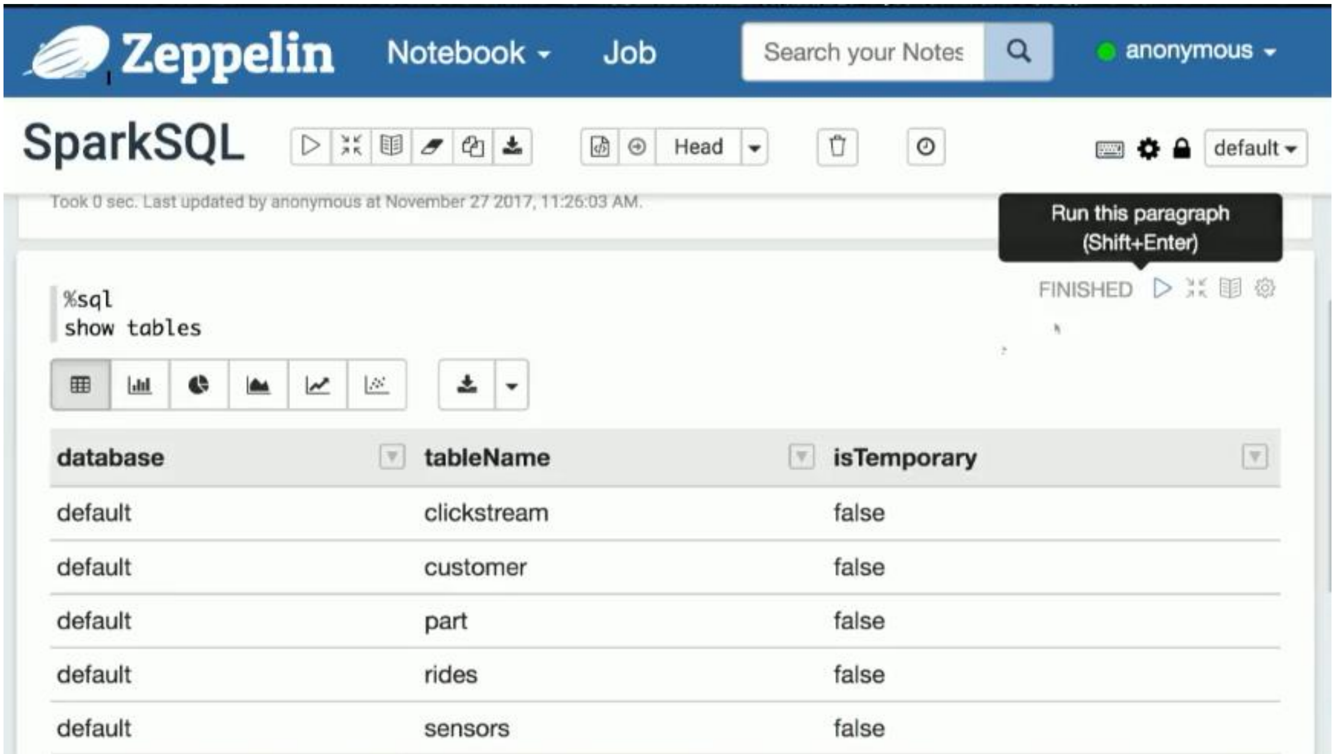


```
%sql
show databases
```

FINISHED

databaseName
buildingadatalake
clickstream
default
jsontesting
monitoring

We can also have our data scientist using Zeppelin notebooks running on the EMR cluster look at the data



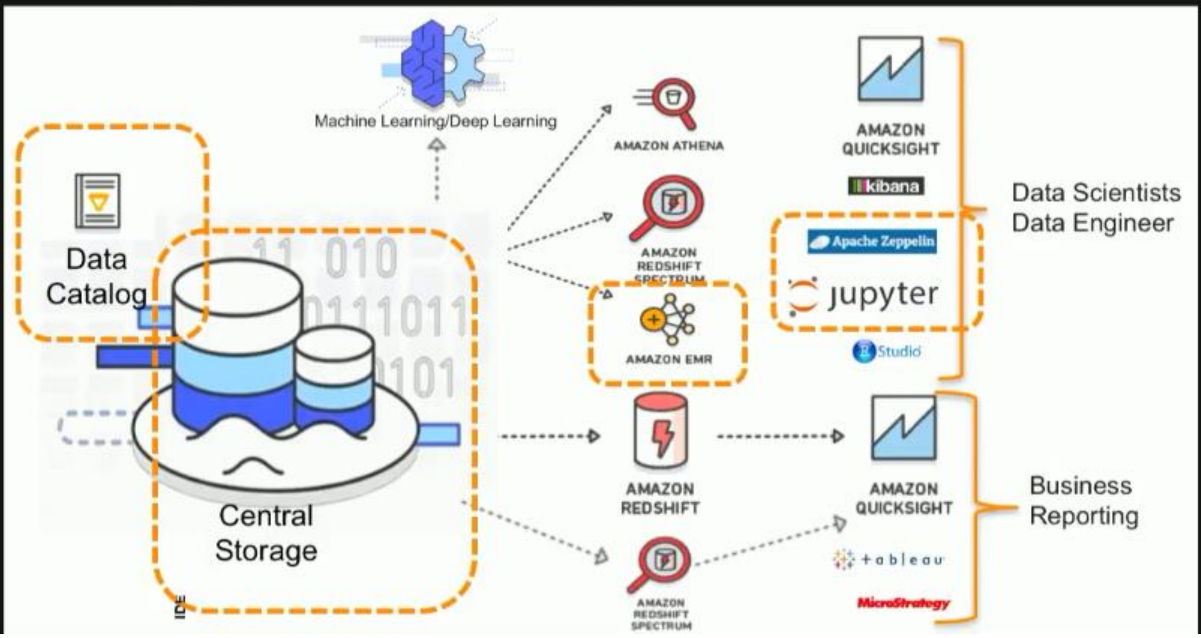
Took 0 sec. Last updated by anonymous at November 27 2017, 11:26:03 AM.

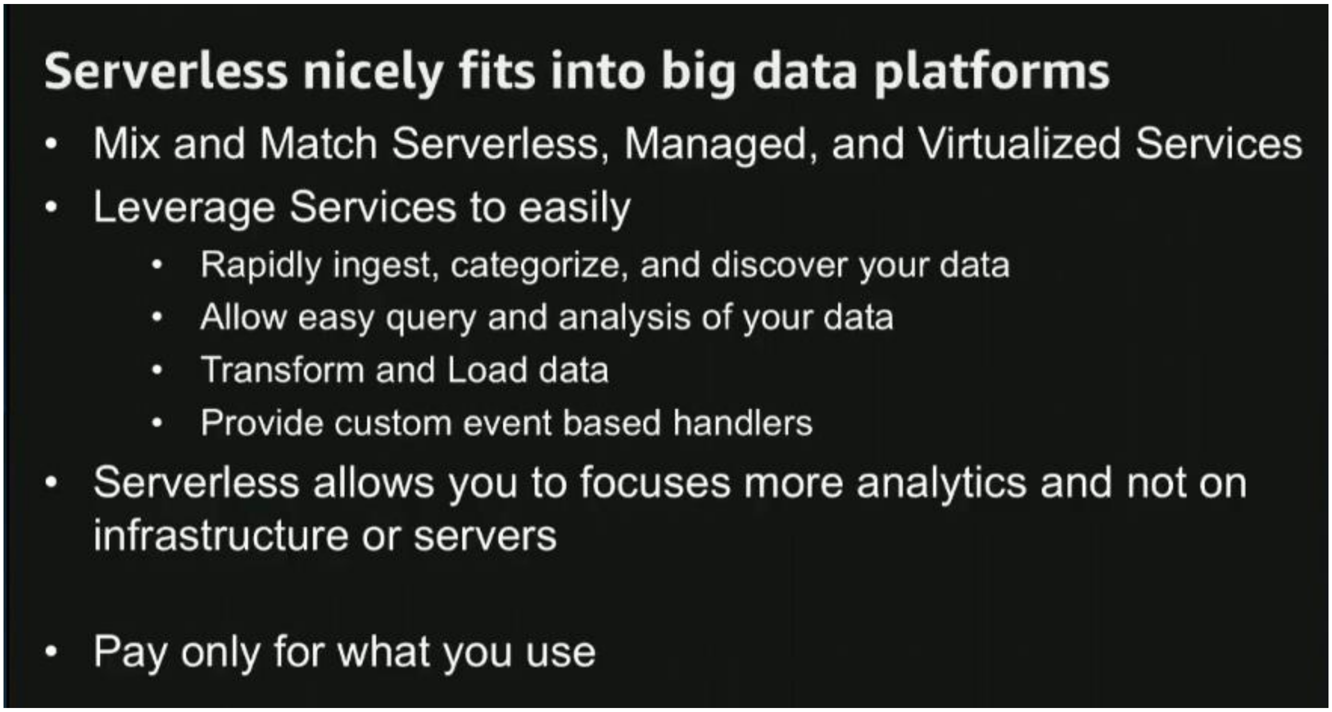
```
%sql
show tables
```

database	tableName	isTemporary
default	clickstream	false
default	customer	false
default	part	false
default	rides	false
default	sensors	false

What about existing Hadoop Clusters?

The Right tool for the Right Job





Serverless nicely fits into big data platforms

- Mix and Match Serverless, Managed, and Virtualized Services
- Leverage Services to easily
  - Rapidly ingest, categorize, and discover your data
  - Allow easy query and analysis of your data
  - Transform and Load data
  - Provide custom event based handlers
- Serverless allows you to focuses more analytics and not on infrastructure or servers
- Pay only for what you use