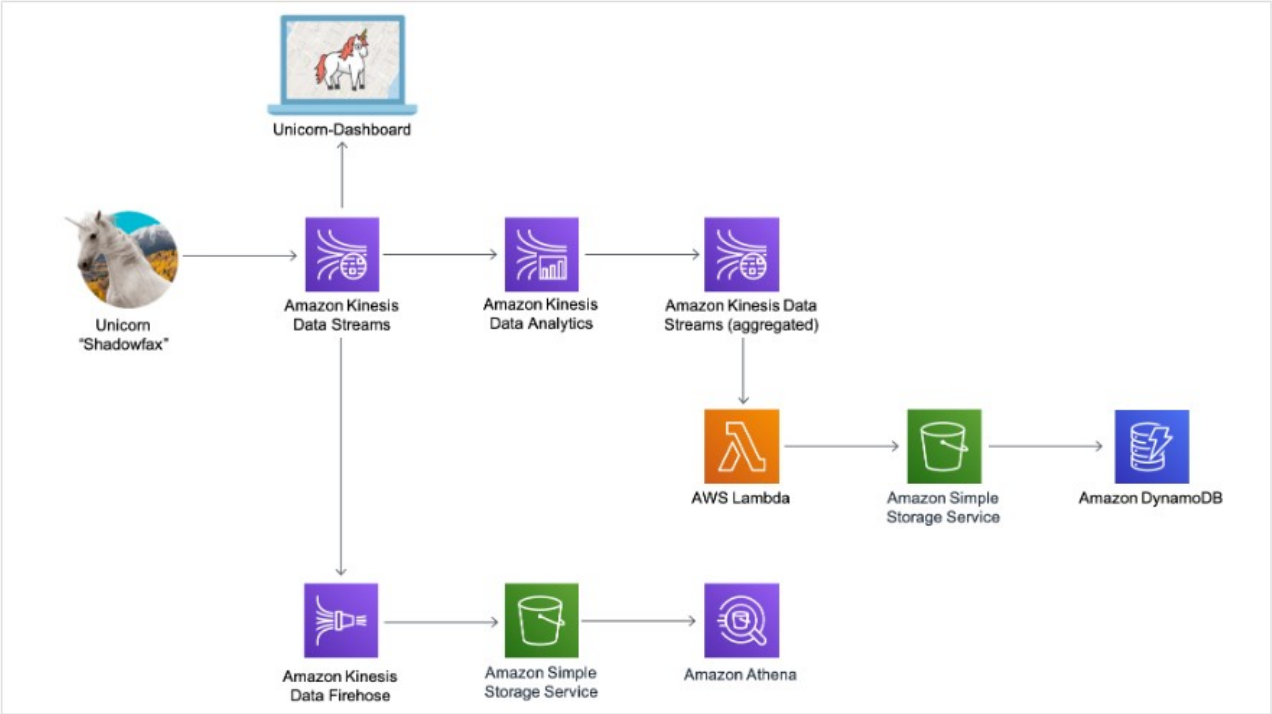
https://aws.amazon.com/getting-started/projects/build-serverless-real-time-data-processing-app-lambda-kinesis-s3-dynamodb-cognito-athena/1/

Build a serverless Real-Time Data Processing App

with AWS Lambda, Amazon Kinesis, Amazon S3, Amazon DynamoDB, Amazon Cognito, and Amazon Athena

Application Architecture



Overview

Serverless applications don't require you to provision, scale, and manage any servers. You can build them for nearly any type of application or backend service, and everything required to run and scale your application with high availability is handled for you.

architectures can be used for many types of applications. For example, you can process transaction orders, analyze click streams, clean data, generate metrics, filter logs, analyze social media, or perform IoT device data telemetry and metering.
In this project, you'll learn how to build a serverless app to process real-time data streams. You'll build infrastructure for a fictional ride-sharing company. In this case, you will enable operations personnel at a fictional Wild Rydes headquarters to monitor the health and status

of their unicorn fleet. Each unicorn is equipped with a sensor that reports its location and vital signs.

You'll use AWS to build applications to process and visualize this data in real-time. You'll use AWS Lambda to process real-time streams, Amazon DynamoDB to persist records in a NoSQL database, Amazon Kinesis Data Analytics to aggregate data, Amazon Kinesis Data Firehose to archive the raw data to Amazon S3, and Amazon Athena to run ad-hoc queries against the raw data.

This workshop is broken up into four modules. You must complete each module before proceeding to the next.

1. Build a data stream

Create a stream in Kinesis and write to and read from the stream to track Wild unicorns on the live map. In this you'll also create an Amazon Cognito identity pool to grant live map access to your stream.

2. Aggregate data

Build a Kinesis Data Analytics application to read from the stream metrics like unicorn health and distance traveled each minute.

3. Process streaming data

Persist aggregate data from the application to a backend database DynamoDB and run queries against those data.

4. Store & query data

Use Kinesis Data Firehose to flush the raw sensor data to an S3 archival purposes. Using Athena, you'll run SQL queries against data for ad-hoc analyses.

Step 4. Create an identity pool for the unicorn dashboard

Create an Amazon Cognito identity pool to grant unauthenticated users access to read from your Kinesis stream. Note the identity pool ID for use in the next step.

---

a. Go to the AWS Management Console, select **Services** then select **Cognito** under Security, Identity & Compliance.
b. Select **Manage Identity Pools.**
c. Select **Create new identity pool.**
d. Enter into **Identity pool name.**
e. Tick the **Enable access to unauthenticated identities** checkbox.
f. Click **Create Pool**.
g. Click **Allow** which will create authenticated and unauthenticated roles for your identity pool.
h. Select **Go to Dashboard.**
i. Select **Edit identity pool** in the upper corner.
j. Note the **Identity pool ID** for use in a later step.
Identity pool ID=us-east-1:425458d5-c9db-4691-b433-fd388df19df2
Identity Pool ARN=arn:aws:cognito-identity:us-east-1:578247465916:identitypool/us-east-1:425458d5-c9db-4691-b433-fd388df19df2

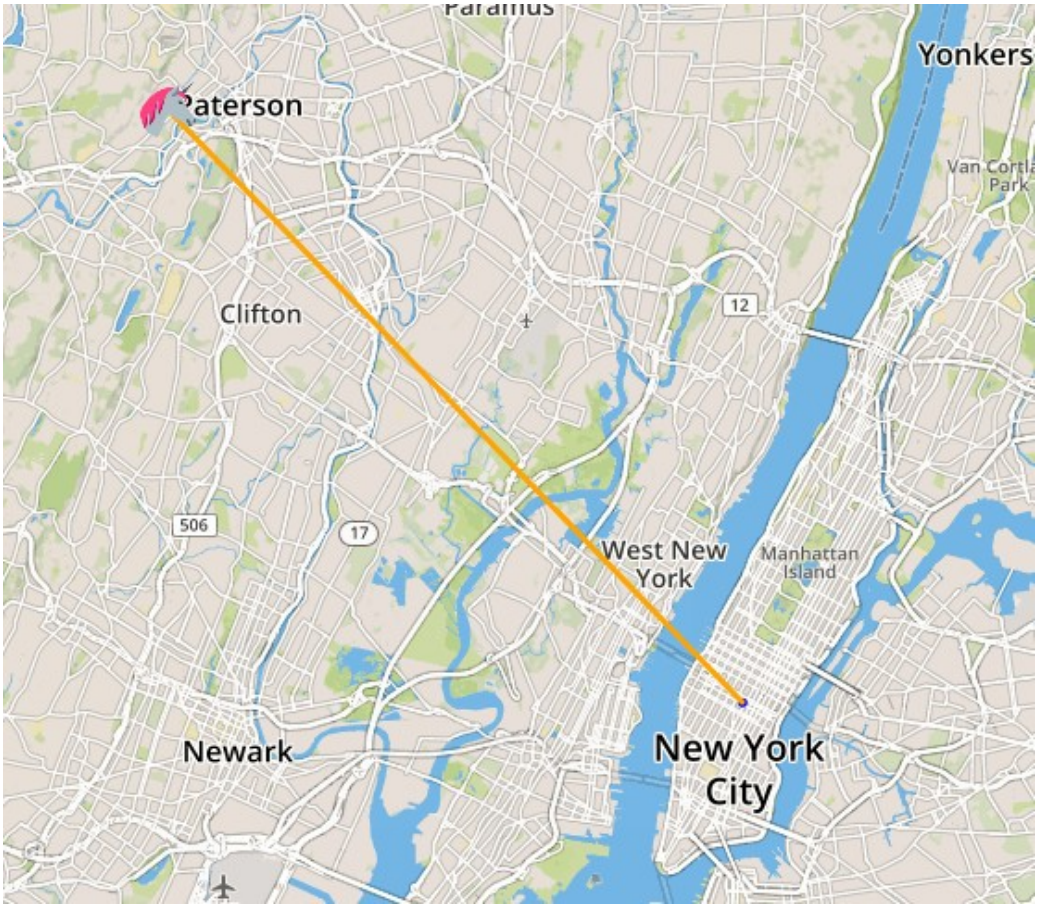
k. Click **Cancel**.

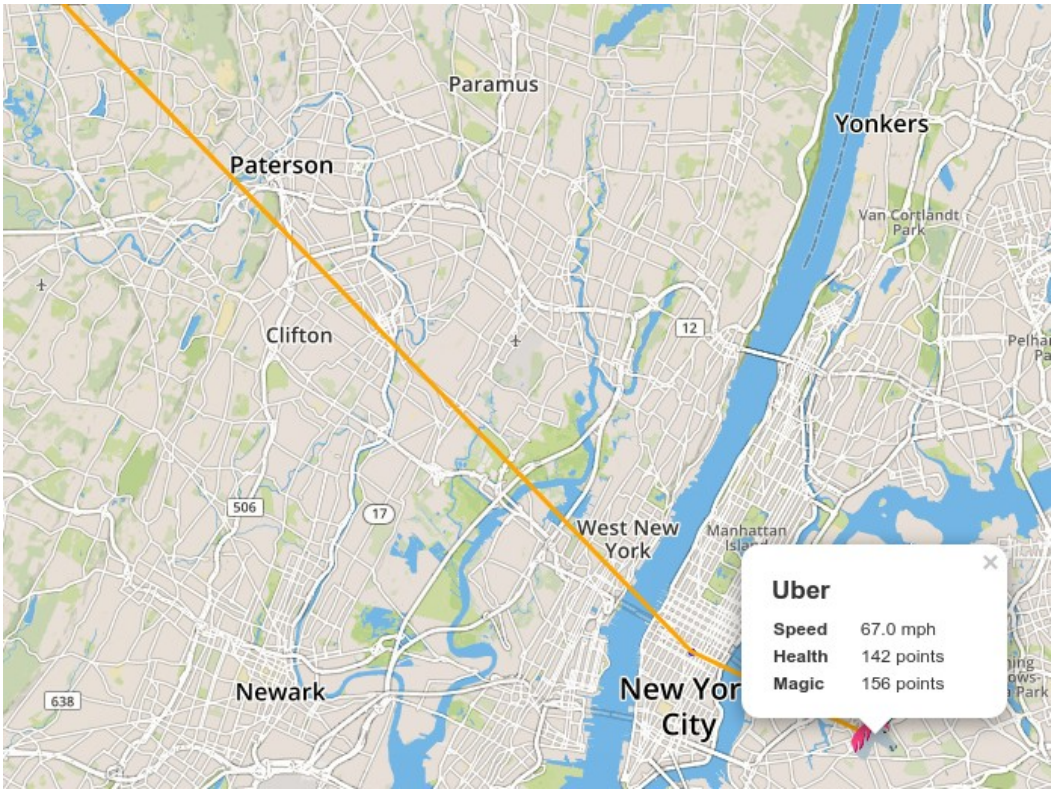
Kinesis stream ARN="arn:aws:kinesis:us-east-1:578247465916:stream/wildrydes"

Dashboard URL=https://dataprocessing.wildrydes.com/dashboard.html

Run a new producer:

```
./producer -name Bucephalus
```


Paramus
Yonkers
Paterson
Van Cortlandt Park
Clifton
12
506
17
West New York
Manhattan Island
Newark
New York City


Paramus
Yonkers
Paterson
Van Cortlandt Park
Clifton
12
506
17
West New York
Manhattan Island
Uber
Speed 67.0 mph
Health 142 points
Magic 156 points
638
Newark
New York City

In this module, you'll create an Amazon Kinesis Data Analytics application to aggregate sensor data from the unicorn fleet in real-time. The application will read from the Amazon Kinesis stream, calculate the total distance traveled, and minimum and maximum health and magic points for each unicorn currently on a Wild Ryde and output these aggregated statistics to an Amazon Kinesis stream every minute.

Step 2. Create an Amazon Kinesis Data Analytics application

Build an Amazon Kinesis Data Analytics application which reads from the **wildrydes** stream built in the previous module and emits a JSON object with the following attributes each minute:

Name	Unicorn name
StatusTime	ROWTIME provided by Amazon Kinesis Data Analytics
Distance	The sum of distance traveled by the unicorn
MinMagicPoints	The data point of the attribute
MaxMagicPoints	The data point of the attribute
MinHealthPoints	The minimum data point of the *HealthPoints* attribute
MaxHealthPoints	The data point of the *HealthPoints* attribute

Application ARN = arn:aws:kinesisanalytics:us-east-1:578247465916:application/wildrydes

Source

Streaming data

Connect to an existing Kinesis stream or Firehose delivery stream, or easily create and c
application can connect to one streaming data source. Learn more

	Source	In-application stream name
	Kinesis stream wildrydes	SOURCE_SQL_STREAM_001

Destination

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

Connect new destination | Disconnect destination

		Destination	In-application stream name
		Kinesis stream wildrydes-summary	DESTINATION_SQL_STREAM

Real time analytics

Continuously analyzing your source data with SQL. Learn more

Go to SQL results

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
  "Name"            VARCHAR(16),
  "StatusTime"        TIMESTAMP,
  "Distance"          SMALLINT,
  "MinMagicPoints"     SMALLINT,
  "MaxMagicPoints"     SMALLINT,
  "MinHealthPoints"    SMALLINT,
  "MaxHealthPoints"    SMALLINT
);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS
```

```
INSERT INTO "DESTINATION_SQL_STREAM"
  SELECT STREAM "Name", "ROWTIME", SUM("Distance"), MIN("MagicPoints"),
            MAX("MagicPoints"), MIN("HealthPoints"), MAX("HealthPoints")
  FROM "SOURCE_SQL_STREAM_001"
  GROUP BY FLOOR("SOURCE_SQL_STREAM_001"."ROWTIME" TO MINUTE), "Name";
```

Source | Real-time analytics | Destination

In-application streams:

- DESTINATION_SQL_STREAM
- error_stream

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

Scroll to bottom when new results arrive.

Filter by column name

ROWTIME	Name	StatusTime	Distance	MinMagicPoints	MaxMagicPoi
2020-02-17 18:56:00.0	Shadowfax	2020-02-17 18:56:00.0	1859	148	154
2020-02-17 18:57:00.0	Shadowfax	2020-02-17 18:57:00.0	1799	148	157
2020-02-17 18:58:00.0	Shadowfax	2020-02-17 18:58:00.0	1769	156	163
2020-02-17 18:59:00.0	Shadowfax	2020-02-17 18:59:00.0	1828	157	162
2020-02-17 19:00:00.0	Shadowfax	2020-02-17 19:00:00.0	1797	155	162
2020-02-17 19:01:00.0	Shadowfax	2020-02-17 19:01:00.0	2970	141	164
2020-02-17 19:02:00.0	Shadowfax	2020-02-17 19:02:00.0	1801	158	165
2020-02-17 19:02:00.0	Uber	2020-02-17 19:02:00.0	866	150	154
2020-02-17 19:03:00.0	Shadowfax	2020-02-17 19:03:00.0	1802	156	167
2020-02-17 19:03:00.0	Uber	2020-02-17 19:03:00.0	1804	148	156

run consumer against summary stream:

```
$ ./consumer -stream "wildrydes-summary"
```


```
{
  "Name": "Shadowfax",
  "StatusTime": "2020-02-17 19:10:00.000",
  "Distance": 1798,
  "MinMagicPoints": 164,
  "MaxMagicPoints": 171,
  "MinHealthPoints": 158,
  "MaxHealthPoints": 164
}{
  "Name": "Uber",
  "StatusTime": "2020-02-17 19:10:00.000",
  "Distance": 1798,
```

"MinMagicPoints": 167,
"MaxMagicPoints": 173,
"MinHealthPoints": 141,
"MaxHealthPoints": 148
}{
"Name": "Shadowfax",
"StatusTime": "2020-02-17 19:11:00.000",
"Distance": 1798,
"MinMagicPoints": 165,
"MaxMagicPoints": 172,
"MinHealthPoints": 162,
"MaxHealthPoints": 170
}

Create DynamoDB table:

name = UnicornSensorData

partition key=*Name*

sort key=*StatusTime*

ARN = arn:aws:dynamodb:us-east-1:578247465916:table/UnicornSensorData

Create IAM policy = arn:aws:iam::578247465916:policy/WildRydesDynamoDBWritePolicy to allow Lambda to write to DynamoDB

{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "VisualEditor0",
"Effect": "Allow",
"Action": "dynamodb:BatchWriteItem",
"Resource": "arn:aws:dynamodb:us-east-1:578247465916:table/UnicornSensorData"
}
]
}

Role ARN = arn:aws:iam::578247465916:role/WildRydesStreamProcessorRole

Attach policies

Policy name	Policy type
AWSLambdaKinesisExecutionRole	AWS managed policy
WildRydesDynamoDBWritePolicy	Managed policy

Create Lambda function (node.js) to process stream

Query DynamoDB

add filter: Name = ‘Uber’

Scan | [Table] UnicornSensorData: Name, StatusTime

Filter | Name | String | = | Uber

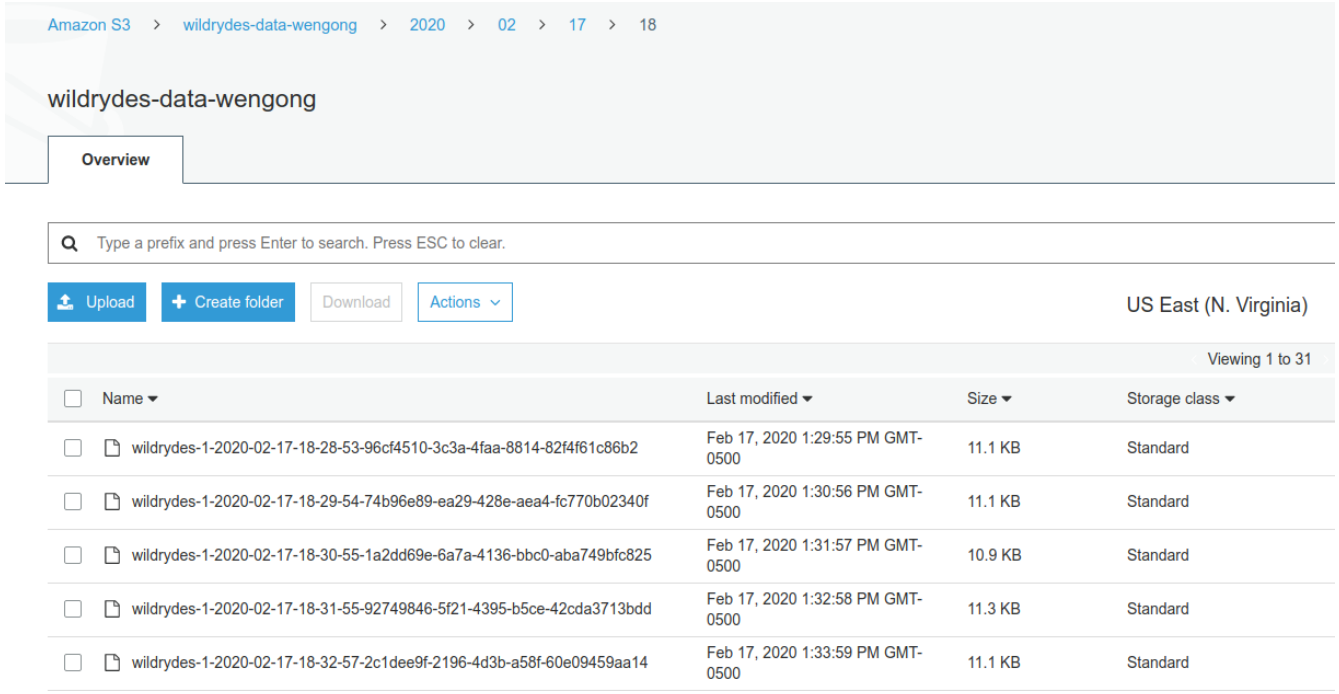
Add filter

Start search

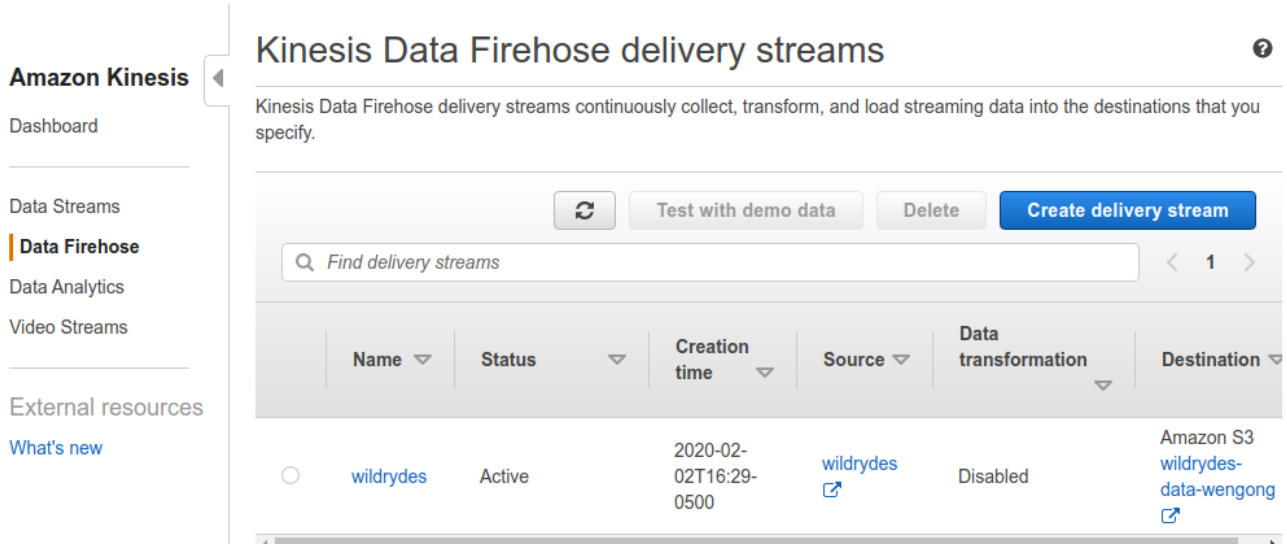
Name	StatusTime	Distance	MaxHealthPoints	MaxMagicPoints	MinHealt
Uber	2020-02-17 19:05:00.000	1799	143	159	137
Uber	2020-02-17 19:06:00.000	1809	148	159	139
Uber	2020-02-17 19:07:00.000	1798	143	161	139
Uber	2020-02-17 19:08:00.000	1832	144	166	137
Uber	2020-02-17 19:09:00.000	1795	149	171	143
Uber	2020-02-17 19:10:00.000	1798	148	173	141
Uber	2020-02-17 19:11:00.000	1797	142	173	137

In this module, you’ll create an Amazon Kinesis Data Firehose to deliver data from the Amazon Kinesis stream created in the first module to Amazon Simple Storage Service (Amazon S3) in batches. You’ll then use Amazon Athena to run queries against our raw data in place.

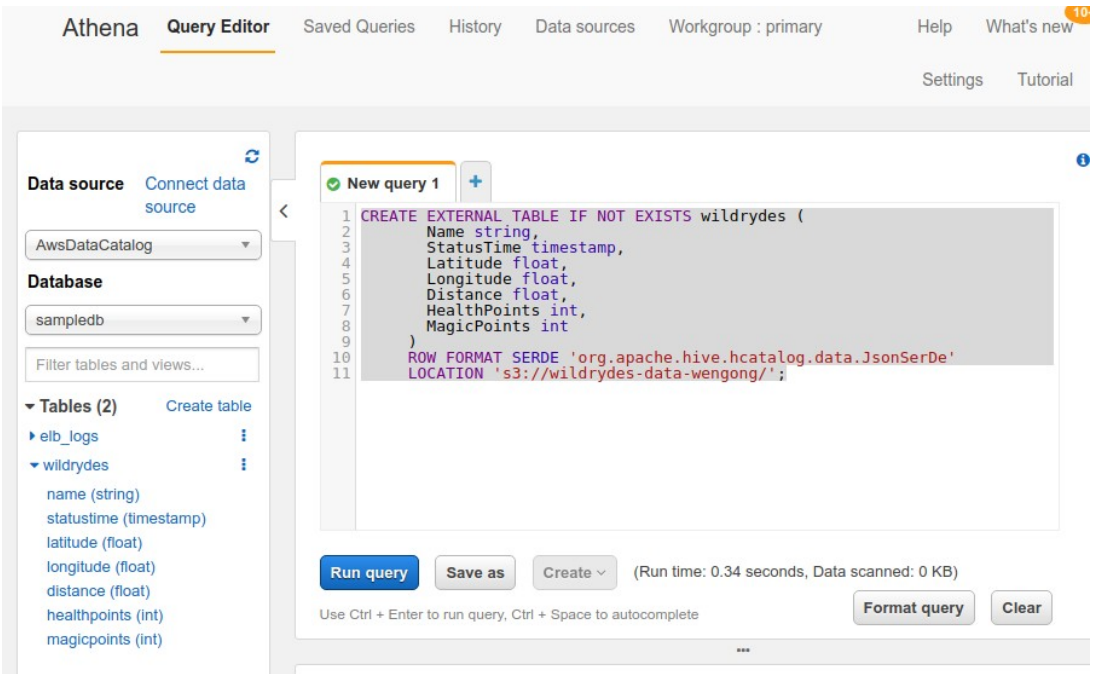
Create S3 bucket to store Firehose data

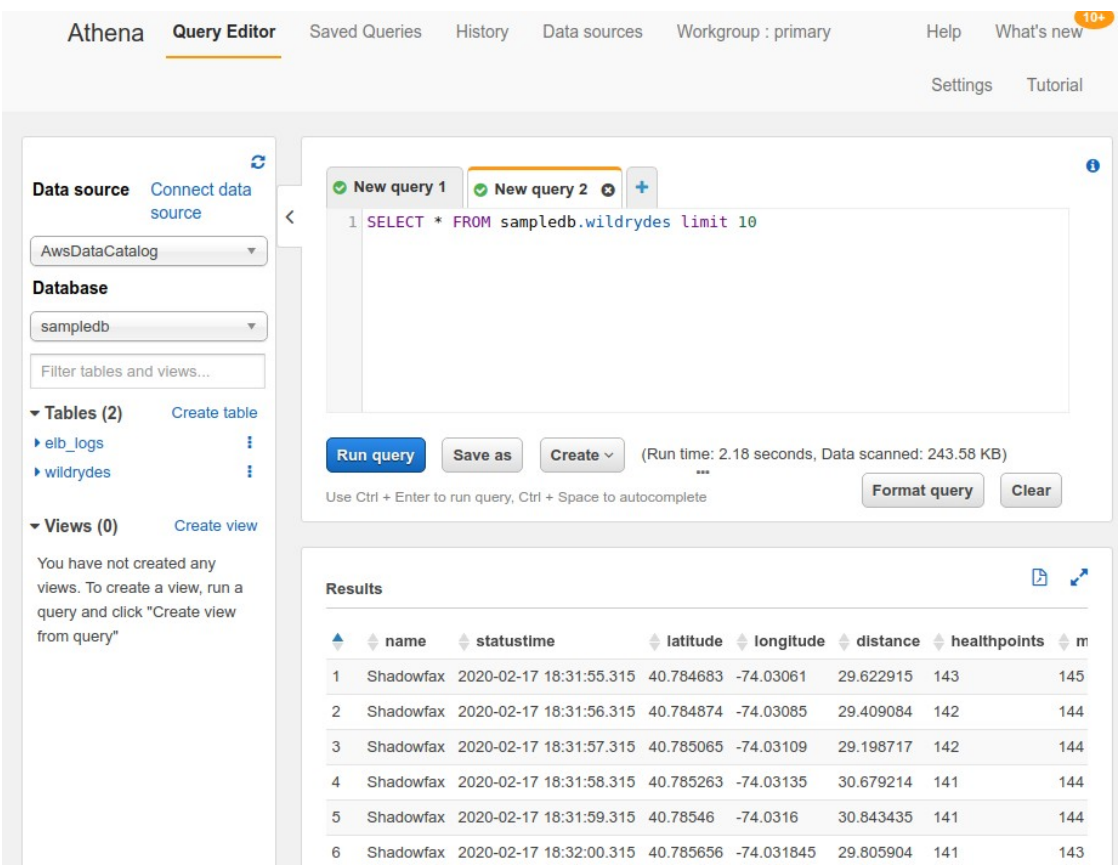


Create Kinesis Firehose



Create Athena table:


Athena
Query Editor
Saved Queries
History
Data sources
Workgroup : primary
Help
What's new
Settings
Tutorial
Data source
Connect data source
AwsDataCatalog
Database
sampledb
Filter tables and views...
Tables (2)
Create table
elb_logs
wildrydes
name (string)
statustime (timestamp)
latitude (float)
longitude (float)
distance (float)
healthpoints (int)
magicpoints (int)
New query 1
CREATE EXTERNAL TABLE IF NOT EXISTS wildrydes (
Name string,
StatusTime timestamp,
Latitude float,
Longitude float,
Distance float,
HealthPoints int,
MagicPoints int
)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
LOCATION 's3://wildrydes-data-wengong/';
Run query
Save as
Create
(Run time: 0.34 seconds, Data scanned: 0 KB)
Use Ctrl + Enter to run query, Ctrl + Space to autocomplete
Format query
Clear


Athena
Query Editor
Saved Queries
History
Data sources
Workgroup : primary
Help
What's new
Settings
Tutorial
Data source
Connect data source
AwsDataCatalog
Database
sampledb
Filter tables and views...
Tables (2)
Create table
elb_logs
wildrydes
Views (0)
Create view
You have not created any views. To create a view, run a query and click "Create view from query"
New query 1
New query 2
SELECT * FROM sampledb.wildrydes limit 10
Run query
Save as
Create
(Run time: 2.18 seconds, Data scanned: 243.58 KB)
Use Ctrl + Enter to run query, Ctrl + Space to autocomplete
Format query
Clear
Results
name
statustime
latitude
longitude
distance
healthpoints
1 Shadowfax 2020-02-17 18:31:55.315 40.784683 -74.03061 29.622915 143 145
2 Shadowfax 2020-02-17 18:31:56.315 40.784874 -74.03085 29.409084 142 144
3 Shadowfax 2020-02-17 18:31:57.315 40.785065 -74.03109 29.198717 142 144
4 Shadowfax 2020-02-17 18:31:58.315 40.785263 -74.03135 30.679214 141 144
5 Shadowfax 2020-02-17 18:31:59.315 40.78546 -74.0316 30.843435 141 144
6 Shadowfax 2020-02-17 18:32:00.315 40.785656 -74.031845 29.805904 141 143

Build a Serverless Web Application with AWS Lambda, Amazon API Gateway, Amazon S3, Amazon DynamoDB, and Amazon Cognito -

https://aws.amazon.com/getting-started/projects/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/