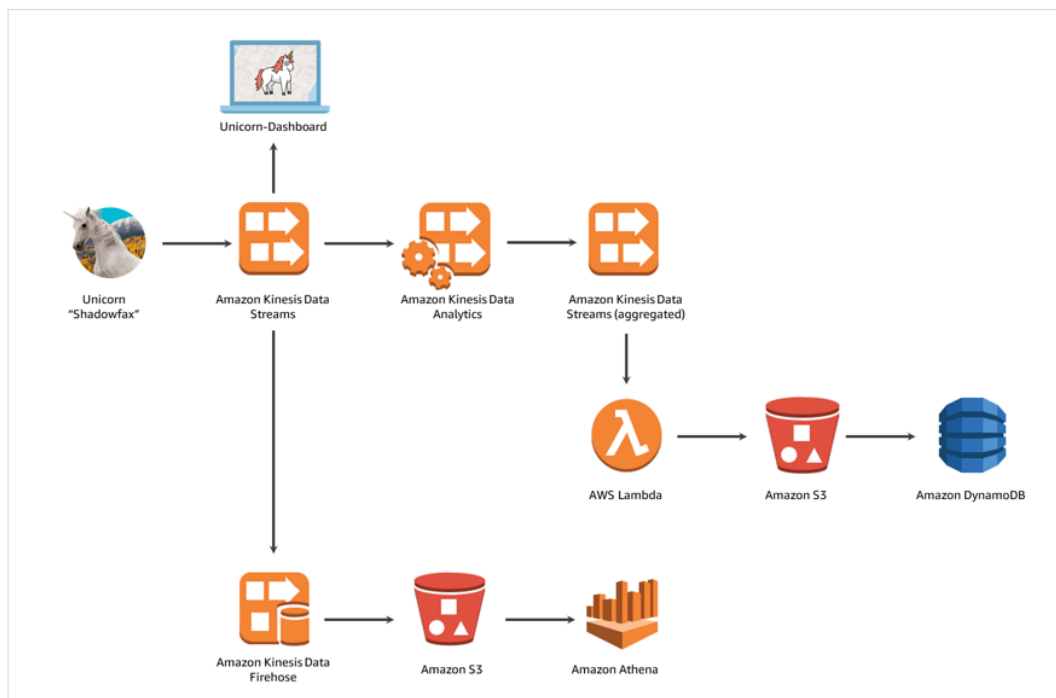


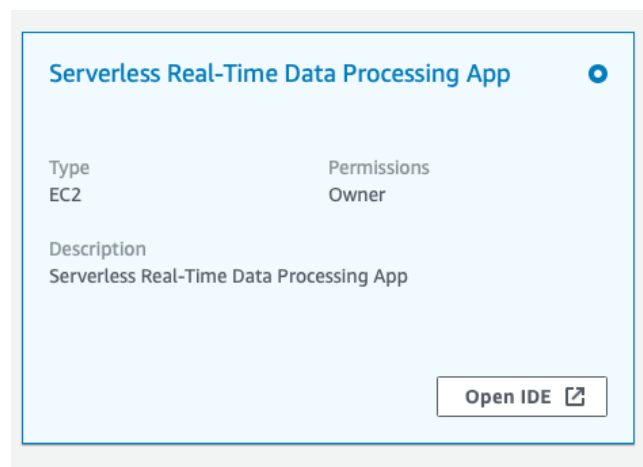
Build a Serverless Real-Time Data Processing App

Intro

Application Architecture



Step 1: Set up your AWS Cloud9 IDE



```
bash - "ip-172-31" x Immediate x (+)
ec2-user:~/environment $ aws sts get-caller identiy
ec2-user:~/environment $ aws sts get-caller identiy
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

    aws help
    aws <command> help
    aws <command> <subcommand> help
aws: error: argument operation: Invalid choice, valid choices are:

assume-role                                | assume-role-with-saml
assume-role-with-web-identity              | decode-authorization-message
get-caller-identity                       | get-federation-token
get-session-token                         | help
ec2-user:~/environment $ █
```

Step 2: Set up the Command Line Clients

Download and unpack the command line clients by running the following command in the Cloud9 terminal:

Build a data stream

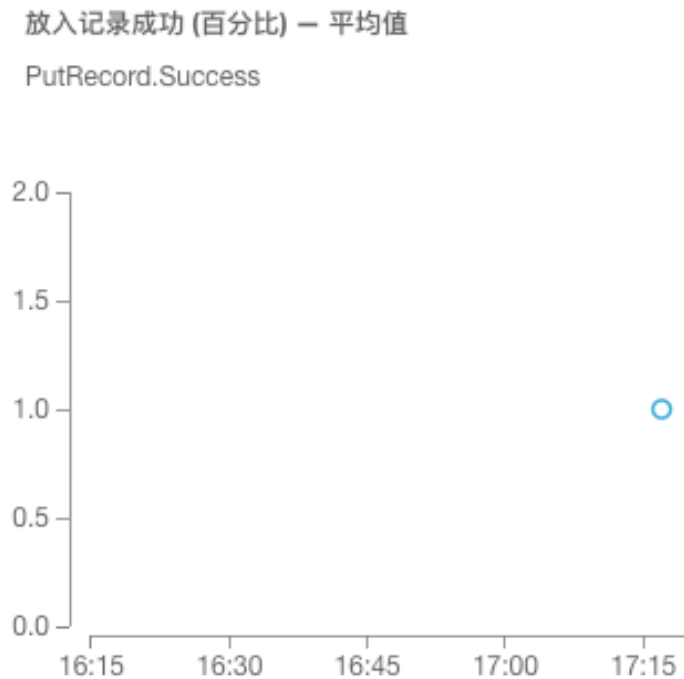
In this module, you'll create an Amazon Kinesis stream to collect and store sensor data from our unicorn fleet.

```
get-session-token | help
ec2-user:~/environment $ curl -s curl -s https://dataprocessing.wildrydes.com/client/client.tar | tar -xv
producer
consumer
ec2-user:~/environment $ █
```

Step 1: Create an Amazon Kinesis stream

<input type="checkbox"/>	Kinesis 流名称	分片数量	状态	使用增强型扇出的用户
<input type="checkbox"/>	wildrydes	1	活跃	0

Step 2: Produce messages into the stream



Step 3: Read messages from the stream

```
./producer - "ip-1 x Immediate x ./consumer - "ip- x (+)
{
  "Distance": 29.57200330703703,
  "HealthPoints": 155,
  "Latitude": 40.69845257476651,
  "Longitude": -74.01016352027426,
  "MagicPoints": 151,
  "Name": "Shadowfax",
  "StatusTime": "2018-11-14 22:20:53.714"
}
{
  "Distance": 29.650897663526155,
  "HealthPoints": 155,
  "Latitude": 40.69820149289851,
  "Longitude": -74.0102871439599,
  "MagicPoints": 151,
  "Name": "Shadowfax",
  "StatusTime": "2018-11-14 22:20:54.714"
}
{
  "Distance": 29.798095978214146,
  "HealthPoints": 154,
  "Latitude": 40.69794916456478,
  "Longitude": -74.0104113808922,
  "MagicPoints": 150,
  "Name": "Shadowfax",
  "StatusTime": "2018-11-14 22:20:55.714"
}
```

Step 4: Create an identity pool for the unicorn dashboard

Edit identity pool

From this page you can modify the details of your identity pool. An identity pool must have a unique name and a set of authenticated and unauthenticated roles. The roles are saved with your identity pool and whenever we receive a request to authorize a user we will automatically utilize the roles you specify here. You will be required to specify the identity pool id from this page when initializing the Amazon Cognito client SDK. [Learn more about using IAM roles with Amazon Cognito.](#)

Identity pool name*

Identity pool ID ⓘ us-east-1:824037b4-f1c0-49a4-a1e7-655049153653 (Show ARN)

Unauthenticated role ⓘ [Create new role](#)

Authenticated role ⓘ [Create new role](#)

Step 5: Grant the unauthenticated role access to the stream

wildrydesDashboardPolicy

Inline policy

✕

Policy summary

{ } JSON

Edit policy

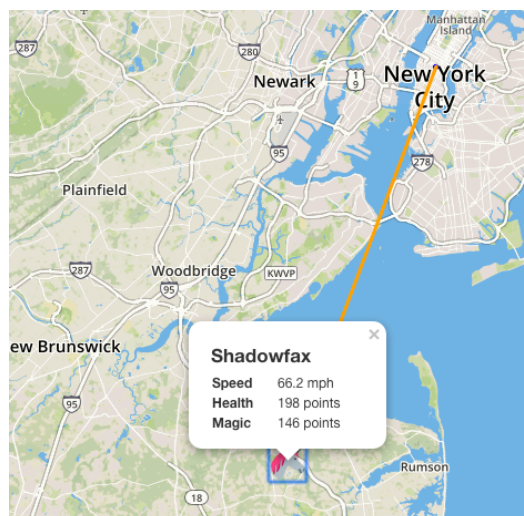
Simulate policy

Q Filter

Service	Access level	Resource	Request condition
Allow (1 of 148 services) Show remaining 147			
Kinesis	Full: List, Read	Multiple	None

▶ Permissions boundary (not set)

Step 6: View unicorn status on the dashboard



Step 7: Experiment with the producer



Aggregate data

In this module, you'll create an Amazon Kinesis Data Analytics application to aggregate sensor data from the unicorn fleet in real-time.

Step 1: Create an Amazon Kinesis stream

<input type="checkbox"/>	wildrydes-summary	1	Active	0
--------------------------	-----------------------------------	---	--------	---

Step 2: Create an Amazon Kinesis Data Analytics application

Real-time analytics

Real-time analytics

Save and run SQL

Add SQL from templates

Download SQL

SQL reference guide

[Kinesis data generator tool](#)

18"MinHealthPoints"SMALLINT,
19"MaxHealthPoints"SMALLINT
20);
21
22CREATE OR REPLACE PUMP "STREAM_PUMP" AS
23INSERT INTO "DESTINATION_SQL_STREAM"
24SELECT STREAM "Name", "ROWTIME", SUM("Distance"), MIN("MagicPoints"),
25MAX("MagicPoints"), MIN("HealthPoints"), MAX("HealthPoints")
26FROM "SOURCE_SQL_STREAM_001"
27GROUP BY FLOOR("SOURCE_SQL_STREAM_001"."ROWTIME" TO MINUTE), "Name";
28

...

Application status: RUNNING

Source data

Real-time analytics

Destination

In-application streams:

Pause results

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

☒ DESTINATION_SQL_STREAM

☐ error_stream

☐ Scroll to bottom when new results arrive.

Filter by column name

ROWTIME	Name	StatusTime	Distance	MinMagicPoints	MaxMagicPoints
2018-11-14 23:22:00.0	Shadowfax	2018-11-14 23:22:00.0	2063	116	122

Connect to a destination

Connect new destination

Disconnect destination

	Destination	In-application stream name	ID
<input type="radio"/>	Kinesis stream wildrydes-summary	DESTINATION_SQL_STREAM	4.1

Step 3: Read messages from the stream

Use the command line consumer to view messages from the Kinesis stream to see the aggregated data being sent every minute.

```
ec2-user:~/environment $ ./consumer -stream wildrydes-summary
{
  "Name": "Shadowfax",
  "StatusTime": "2018-11-14 23:27:00.000",
  "Distance": 1794,
  "MinMagicPoints": 118,
  "MaxMagicPoints": 125,
  "MinHealthPoints": 123,
  "MaxHealthPoints": 129
}
{
  "Name": "Bucephalus",
  "StatusTime": "2018-11-14 23:27:00.000",
  "Distance": 1803,
  "MinMagicPoints": 134,
  "MaxMagicPoints": 142,
  "MinHealthPoints": 157,
  "MaxHealthPoints": 166
}
```

Step 4: Experiment with the producer

Multiple unicorn

```
./producer - "ip-1 x Immediate x ./consi
  "MinHealthPoints": 123,
  "MaxHealthPoints": 129
}
{
  "Name": "Bucephalus",
  "StatusTime": "2018-11-14 23:27:00.000",
  "Distance": 1803,
  "MinMagicPoints": 134,
  "MaxMagicPoints": 142,
  "MinHealthPoints": 157,
  "MaxHealthPoints": 166
}
{
  "Name": "Shadowfax",
  "StatusTime": "2018-11-14 23:28:00.000",
  "Distance": 1797,
  "MinMagicPoints": 116,
  "MaxMagicPoints": 127,
  "MinHealthPoints": 126,
  "MaxHealthPoints": 133
}
{
  "Name": "Bucephalus",
  "StatusTime": "2018-11-14 23:28:00.000",
  "Distance": 1800,
  "MinMagicPoints": 134,
  "MaxMagicPoints": 142,
  "MinHealthPoints": 161,
  "MaxHealthPoints": 167
}
```

Process streaming data

Step 1: Create an Amazon DynamoDB tables

Create tableDelete table

Filter by table name

Name

UnicornSensorData

UnicornSensorDataClose

OverviewItemsMetricsAlarmsCapacityIndexesGlobal TablesMore

Recent alerts

No CloudWatch alarms have been triggered for this table.

Stream details

Stream enabledNo

View type-

Latest stream ARN-

Manage Stream

Table details

Table nameUnicornSensorData

Primary partition keyName (String)

Primary sort keyStatusTime (String)

Point-in-time recoveryDISABLEDEnable

EncryptionDISABLED

Time to live attributeDISABLEDManage TTL

Table statusActive

Creation dateNovember 14, 2018 at 6:30:47 PM UTC-5

Provisioned read capacity units5 (Auto Scaling Disabled)

Provisioned write capacity units5 (Auto Scaling Disabled)

Last decrease time-

Last increase time-

Storage size (in bytes)0 bytes

Item count0

RegionUS East (N. Virginia)

Amazon Resource Name (ARN)arn:aws:dynamodb:us-east-1:747142172942:table/UnicornSensorData

Storage size and item count are not updated in real-time. They are updated periodically, roughly every six hours.

Step 2: Create an IAM role for your Lambda function

Create policy

Policy ARNarn:aws:iam::747142172942:policy/WildRydesDynamoDBWritePolicy

Description

PermissionsPolicy usagePolicy versionsAccess Advisor

Policy summary{} JSONEdit policy

Filter

Service	Access level	Resource	Request condition
Allow (1 of 148 services) Show remaining 147			
DynamoDB	Limited: Write	TableName string like UnicornSensorData	None

Create role

Summary

Delete role

Role ARN	arn:aws:iam::747142172942:role/WildRydesStreamProcessorRole
Role description	Allows Lambda functions to call AWS services on your behalf. Edit
Instance Profile ARNs	
Path	/
Creation time	2018-11-14 18:36 EST
Maximum CLI/API session duration	1 hour Edit

Permissions

Trust relationships

Access Advisor

Revoke sessions

▼ Permissions policies (2 policies applied)

Attach policies

Add inline policy

Policy name ▼	Policy type ▼	
WildRydesDynamoDBWritePolicy	Managed policy	✕
AWSLambdaKinesisExecutionRole	AWS managed policy	✕

▶ Permissions boundary (not set)

Step 3: Create a Lambda function to process the stream

▼ Designer

CloudFront

CloudWatch Events

CloudWatch Logs

CodeCommit

Cognito Sync Trigger

DynamoDB

Kinesis

S3

SNS

SQS

WildRydesStreamProcessor

✓ Saved

Kinesis

✓ Saved

Add triggers from the list on the left

Amazon CloudWatch Logs

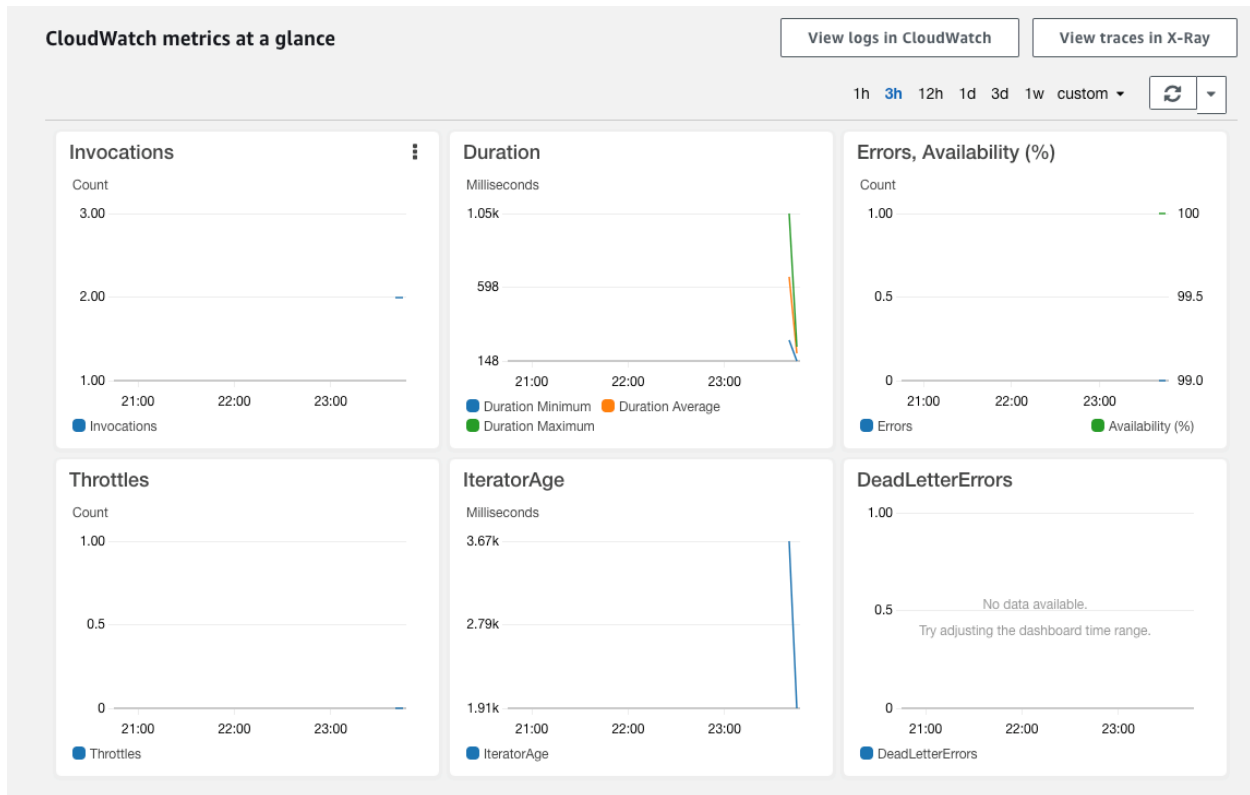
Amazon DynamoDB

Amazon Kinesis

Resources that the function's role has access to appear here

Kinesis

Step 4: Monitor the Lambda function



Step 5: Query the DynamoDB table

Store & query data

Dashboard

- Tables
- Backups
- Reserved capacity
- Preferences

DAX

- Dashboard
- Clusters
- Subnet groups
- Parameter groups
- Events

Filter by table name

Name

UnicornSensorData

Overview Items Metrics Alarms Capacity Indexes Global Tables More

Create item Actions

Scan: [Table] UnicornSensorData: Name, StatusTime

Viewing 1 to 33 items

Scan [Table] UnicornSensorData: Name, StatusTime

Add filter

Start search

	Name	StatusTime	Distance	MaxHealthPoint	MaxMagicPoint
<input type="checkbox"/>	Bucephalus	2018-11-14 23:43:00.000	1803	134	129
<input type="checkbox"/>	Bucephalus	2018-11-14 23:44:00.000	538	131	125
<input type="checkbox"/>	Rocinante	2018-11-14 23:44:00.000	814	150	154
<input type="checkbox"/>	Rocinante	2018-11-14 23:45:00.000	1796	150	154
<input type="checkbox"/>	Rocinante	2018-11-14 23:46:00.000	1803	149	160
<input type="checkbox"/>	Rocinante	2018-11-14 23:47:00.000	1797	149	153
<input type="checkbox"/>	Rocinante	2018-11-14 23:48:00.000	1774	155	149
<input type="checkbox"/>	Rocinante	2018-11-14 23:49:00.000	1830	160	145
<input type="checkbox"/>	Rocinante	2018-11-14 23:50:00.000	1804	161	137
<input type="checkbox"/>	Rocinante	2018-11-14 23:51:00.000	1795	156	136
<input type="checkbox"/>	Rocinante	2018-11-14 23:52:00.000	1829	160	135
<input type="checkbox"/>	Rocinante	2018-11-14 23:53:00.000	1802	157	135
<input type="checkbox"/>	Rocinante	2018-11-14 23:54:00.000	1799	156	136

Step 1: Create an Amazon S3 bucket

+ Create bucket	Empty	Delete	4 Buckets	0 Public	1 Regions	↻
Bucket name ↑ ≡	Access i ↑ ≡	Region ↑ ≡	Date created ↑ ≡			
aws-logs-747142172942-us-east-1	Not public *	US East (N. Virginia)	Nov 11, 2018 6:47:12 PM GMT-0500			
test950607	Not public *	US East (N. Virginia)	Nov 11, 2018 5:45:38 PM GMT-0500			
us-east-1.elasticmpreduce.samples	Not public *	US East (N. Virginia)	Nov 11, 2018 8:27:22 PM GMT-0500			
wildehydes-data-yi	Not public *	US East (N. Virginia)	Nov 14, 2018 7:13:33 PM GMT-0500			

* Objects might still be publicly accessible due to object ACLs. [Learn more](#)

Step 2: Create an Amazon Kinesis Data Firehose delivery stream

Kinesis Firehose delivery streams

Kinesis Firehose delivery streams continuously collect, transform, and load streaming data into the destinations that you specify.

[Create delivery stream](#) [Test with demo data](#) [Delete](#)

Filter or search by name

Viewing 1 - 1 of 1 delivery streams

Name	Status	Created	Source	Record transformation i	Destination
wildrydes	Active	2018-11-14T19:22-0500	wildrydes	Disabled	Amazon S3 wildehydes-data-yi

Viewing 1 - 1 of 1 delivery streams

Step 3: Create an Amazon Athena table

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

```
1 CREATE EXTERNAL TABLE IF NOT EXISTS wildrydes (  
2   Name string,  
3   StatusTime timestamp,  
4   Latitude float,  
5   Longitude float,  
6   Distance float,  
7   HealthPoints int,  
8   MagicPoints int  
9 )  
10 ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'  
11 LOCATION 's3://wildehydes-data-yi/';
```

[Run query](#) [Save as](#) [Create](#) (Run time: 0.4 seconds, Data scanned: 0KB) [Format query](#) [Clear](#)

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

Results

Query successful.

Step 4: Explore the batched data files

```
wildrydes-1-2018-11-15-00-22-25-20b18066-00e5-45bc-baba-0dd
{"Distance":30.464808350996236,"HealthPoints":118,"Latitude":
40.12478220215499,"Longitude":-73.84456080840864,"MagicPoints":
99,"Name":"Rocinante","StatusTime":"2018-11-15 00:22:25.298"}
{"Distance":29.014347988024166,"HealthPoints":118,"Latitude":
40.1245235494687,"Longitude":-73.84450282416408,"MagicPoints":
99,"Name":"Rocinante","StatusTime":"2018-11-15 00:22:26.298"}
{"Distance":29.573175731910727,"HealthPoints":118,"Latitude":
40.12425991503003,"Longitude":-73.84444372334505,"MagicPoints":
99,"Name":"Rocinante","StatusTime":"2018-11-15 00:22:27.298"}
{"Distance":29.666279000971944,"HealthPoints":118,"Latitude":
40.12399545060853,"Longitude":-73.84438443669275,"MagicPoints":
98,"Name":"Rocinante","StatusTime":"2018-11-15 00:22:28.298"}
{"Distance":29.170044780222568,"HealthPoints":118,"Latitude":
40.12373540994022,"Longitude":-73.84432614196778,"MagicPoints":
98,"Name":"Rocinante","StatusTime":"2018-11-15 00:22:29.298"}
{"Distance":30.005435980604958,"HealthPoints":118,"Latitude":
40.12346792205378,"Longitude":-73.84426617798897,"MagicPoints":
99,"Name":"Rocinante","StatusTime":"2018-11-15 00:22:30.298"}
{"Distance":29.89237217951097,"HealthPoints":118,"Latitude":
40.12320144209128,"Longitude":-73.8442064401961,"MagicPoints":
100,"Name":"Rocinante","StatusTime":"2018-11-15 00:22:31.298"}
{"Distance":29.00255449684717,"HealthPoints":119,"Latitude":
40.122942894539804,"Longitude":-73.84414848086816,"MagicPoints":
100,"Name":"Rocinante","StatusTime":"2018-11-15 00:22:32.298"}
{"Distance":29.292295325051494,"HealthPoints":119,"Latitude":
40.12268176405097,"Longitude":-73.84408994273853,"MagicPoints":
101,"Name":"Rocinante","StatusTime":"2018-11-15 00:22:33.298"}
{"Distance":29.902615249183796,"HealthPoints":119,"Latitude":
40.12241519277511,"Longitude":-73.84403018516663,"MagicPoints":
101,"Name":"Rocinante","StatusTime":"2018-11-15 00:22:34.298"}
*****
```

Step 5: Query the data file

The screenshot shows a database management interface. On the left, the 'Database' sidebar shows a selected database 'sampledb' with tables 'elb_logs' and 'wildrydes'. The main area displays a 'New query 1' editor with the SQL query: `SELECT * FROM wildrydes`. Below the editor, buttons for 'Run query', 'Save as', and 'Create' are visible, along with a status bar indicating 'Run time: 1.63 seconds, Data scanned: 163.47KB'. The 'Results' section at the bottom shows a table with 7 columns: name, statustime, latitude, longitude, distance, healthpoints, and magicpoints. It contains 4 rows of data for 'Rocinante'.

	name	statustime	latitude	longitude	distance	healthpoints	magicpoints
1	Rocinante	2018-11-15 00:33:01.298	39.954636	-73.806465	30.327583	93	108
2	Rocinante	2018-11-15 00:33:02.298	39.95436	-73.806404	30.898773	92	109
3	Rocinante	2018-11-15 00:33:03.298	39.954098	-73.80634	29.506586	91	109
4	Rocinante	2018-11-15 00:33:04.298	39.953823	-73.80628	30.999794	91	110