



## Workshop

### **Building Your First Big Data Application on AWS**

## Table of Contents

<b>PREREQUISITES.....</b>	<b>4</b>
<b>WORKSHOP OVERVIEW AND BIG DATA APPLICATION ARCHITECTURE .....</b>	<b>5</b>
<b>WORKSHOP SCRIPTS .....</b>	<b>5</b>
<b>LAUNCH CLOUDFORMATION TEMPLATE TO SETUP ENVIRONMENT .....</b>	<b>6</b>
<b>TASK 1: COLLECT LOGS W/A KINESIS DATA FIREHOSE DELIVERY STREAM (5-MIN).....</b>	<b>9</b>
1.1 VERIFY LAMBDA FUNCTION DELIVERING LOGS .....	9
1.2 MONITORING KINESIS DATA FIREHOSE DELIVERY TO AMAZON S3 .....	10
<b>TASK 2: REAL-TIME DATA PROCESSING USING AMAZON KINESIS DATA ANALYTICS (20 MIN) .....</b>	<b>12</b>
2.1 START AMAZON KINESIS DATA ANALYTICS APP .....	12
2.1 CALCULATE AN AGGREGATE METRIC.....	15
2.2 ANOMALY DETECTION.....	15
2.3 KINESIS DATA ANALYTICS IN-APPLICATION STREAMS .....	16
<b>TASK 3: DELIVER STREAMING RESULTS TO AMAZON REDSHIFT USING KINESIS DATA FIREHOSE (5 MIN).....</b>	<b>17</b>
3.1 CONNECT TO AMAZON REDSHIFT CLUSTER .....	17
3.2 CREATE TABLE IN AMAZON REDSHIFT.....	18
3.3 DELIVER DATA TO AMAZON REDSHIFT USING DATA FIREHOSE.....	19
3.4 AMAZON REDSHIFT - TEST QUERIES .....	20
<b>TASK 4: TRANSFORM WEBLOGS TO PARQUET FORMAT USING AWS GLUE (30 MIN) .....</b>	<b>22</b>
4.1 DISCOVER DATASET WITH AWS GLUE.....	22
4.2 CREATE ETL JOB IN AWS GLUE .....	27
<b>TASK 5: QUERYING AMAZON S3 DATA USING REDSHIFT SPECTRUM AND AMAZON ATHENA (30 MIN) .....</b>	<b>32</b>
5.1 SET UP AWS GLUE CRAWLER FOR PROCESSED PARQUET DATA.....	32
5.2 QUERY USING AMAZON REDSHIFT SPECTRUM .....	34
5.3 QUERYING USING AMAZON ATHENA .....	35

<b>TASK 6 (OPTIONAL): DATA VISUALIZATION WITH AMAZON QUICKSIGHT (20 MIN).....</b>	<b>36</b>
6.1    AMAZON QUICKSIGHT REGISTRATION.....	37
6.2    CONNECT TO AMAZON REDSHIFT (DATA STORE) FOR QUICKSIGHT .....	38
6.3    CREATING YOUR FIRST ANALYSIS .....	41
<b>TASK 7 (OPTIONAL): INTERACTIVE ANALYSIS USING AMAZON EMR (20 MIN).....</b>	<b>42</b>
7.1    OPEN THE ZEPPELIN INTERFACE .....	42
7.2    RUN THE NOTEBOOK.....	43
<b>DELETE ALL RESOURCES.....</b>	<b>46</b>

# Prerequisites

In order to complete this workshop, you'll need

- An active AWS Account,
- An AWS IAM user in that account with at least full permissions to the following AWS services:
  - AWS IAM
  - Amazon S3
  - Amazon Kinesis
  - AWS Glue
  - Amazon Redshift
  - Amazon SageMaker

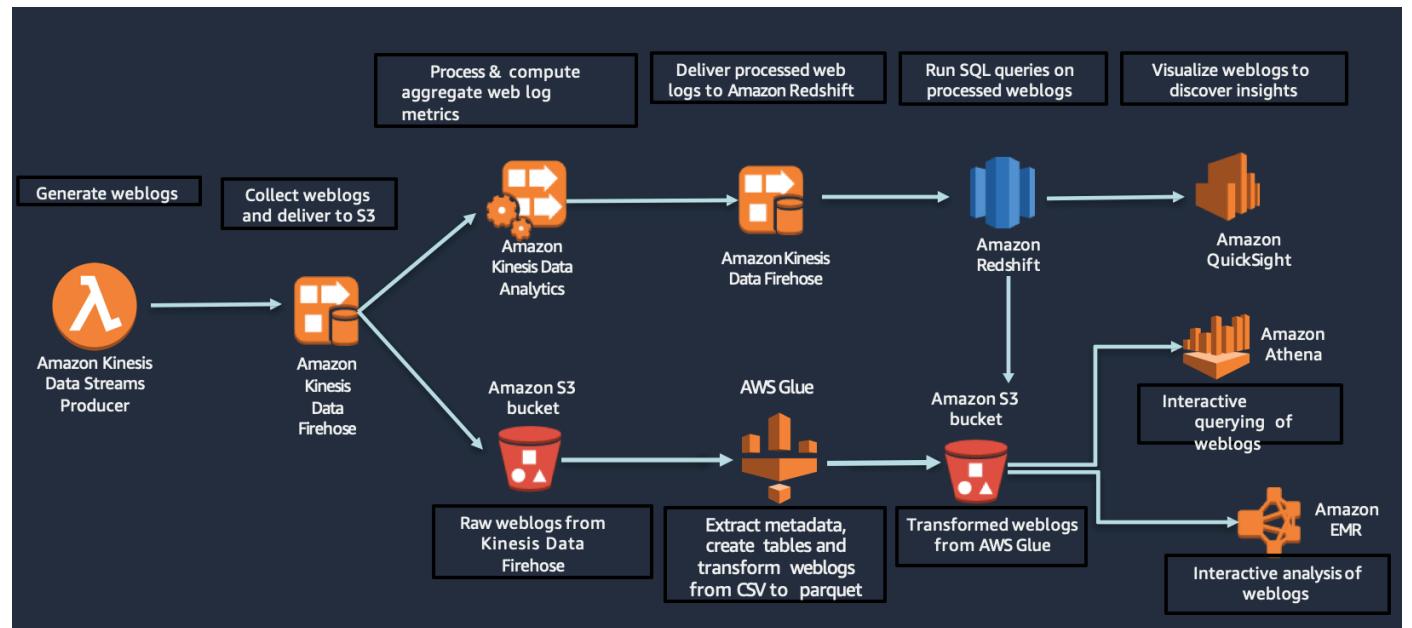
**If you use your OWN Account:** The code and instructions in this workshop can support multiple students in a given AWS account. If you try sharing an account with another student, you'll might run into naming conflicts for certain resources if launching with the existing VPC stack. If you run into this conflict please launch with new VPC keeping in mind soft limits for the number of VPC's that can be created within a region.

**Use a personal account or create a new AWS account, created/activated 24 hours ago, for this workshop** rather than using an organization's account to ensure you have full access to the necessary services and to ensure you do not leave behind any resources from the workshop.

**Costs:** Some, but NOT all, of the resources you will launch as part of this workshop are eligible for the AWS free tier if your account is less than 12 months old. See the [AWS Free Tier page](#) for more details. An example of a resource that is **not** covered by the free tier is the ml.m4.xlarge notebook instance used in some workshops. To avoid charges for endpoints and other resources you might not need after you've finished a workshop.

# Workshop Overview and Big Data Application Architecture

This workshop demonstrates use of AWS Data & Analytics services, such as, Amazon S3, Amazon Kinesis, Amazon Redshift, AWS Glue, and Amazon SageMaker to build a big data application. It does so via a set of straightforward examples for common use cases including real-time streaming, building your Data Lake catalog, and processing the data with a number of Analytics engines.



## Workshop Scripts

- Download **BigDataWorkshop.zip**, if not already done,  
Location: <https://github.com/wrbaldwin/da-week/tree/master/Labs/First-Big-Data-App>  
Or  
Use direct link: [Workshop Scripts](#)
- Unzip file **BigDataWorkshop.zip**.
- You will need these scripts during workshop.

# Launch CloudFormation template to setup environment

In this lab, you will first create AWS resources needed for the workshop using a CloudFormation template.

1. You can launch this CloudFormation stack in your account in one of the supported Regions below. Please click on "Launch Stack" based on your choice.

AWS Region	Short name	New VPC	Existing VPC
US East (N. Virginia)	us-east-1	<a href="#">Launch Stack</a>	<a href="#">Launch Stack</a>
US West (Oregon)	us-west-2	<a href="#">Launch Stack</a>	<a href="#">Launch Stack</a>
EU (Ireland)	eu-west-1	<a href="#">Launch Stack</a>	<a href="#">Launch Stack</a>
EU (Frankfurt)	eu-central-1	<a href="#">Launch Stack</a>	<a href="#">Launch Stack</a>

CloudFormation > Stacks > Create stack

Step 1  
Specify template

Step 2  
Specify stack details

Step 3  
Configure stack options

Step 4  
Review

Create stack

Prerequisite - Prepare template

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

Template is ready    Use a sample template    Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL    Upload a template file

Amazon S3 URL

Amazon S3 template URL

[View in Designer](#)

Cancel   [Next](#)

2. Verify the Amazon Simple Storage Service (Amazon S3) template URL, and click "Next"
3. Enter Stack Name: **bd-workshop** (if multiple users are using same account and same region – include suffice to keep name unique)
4. Enter other parameters as shown below:
  - a. All defaults can be left as-is for this workshop.
  - b. Depending on if you selected new or existing, the only change is picking a CIDR range for the VPC or selecting an existing VPC in your account.
  - c. If there is a Amazon Redshift password in there, leave as is. If there is no password, enter

Abc12345 as the password. Make a note of this for later.

## Specify stack details

### Stack name

Stack name

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

### Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

#### Redshift Configuration

UserName

Password

Default value is 'Abc12345'. Include 1 Upper, 1 Lower, 1 Number, min 8 characters

DatabaseName

NodeType

The type of Redshift node to be provisioned



#### Network Configuration

VpcCIDR

Please enter the IP range (CIDR notation) for the VPC to be created

PublicSubnetCIDR

Please enter the IP range (CIDR notation) for the public subnet

#### Environment Configuration

EnvironmentName

An environment name that will be prefixed to resource names

IncludeRedshift

Specifies whether to include the Redshift section of the workshop.



IncludeEMR

Specifies whether to include the EMR section of the workshop.



IncludeSageMaker

Specifies whether to include the SageMaker section of the workshop.

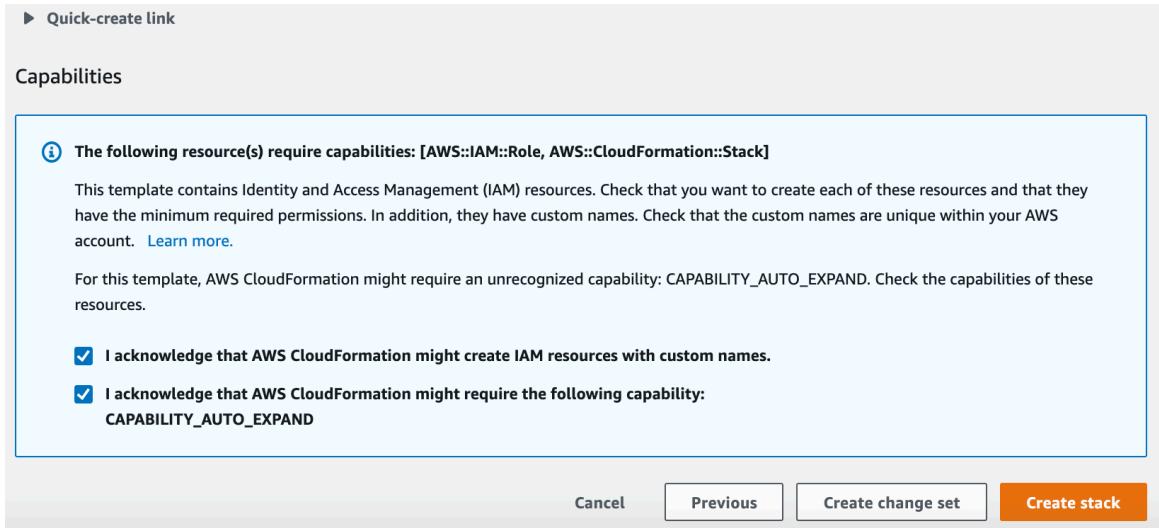


Cancel

Previous

Next

5. Click "Next"
6. Leave the Options at their defaults and click "Next".
7. On the **Review** page, at the bottom of the screen please make sure you
8. Check the box '**I acknowledge that AWS CloudFormation might create IAM resources with custom names.**'
9. Check the box '**I acknowledge that AWS CloudFormation might require the following capability CAPABILITY\_AUTO\_EXPAND'**



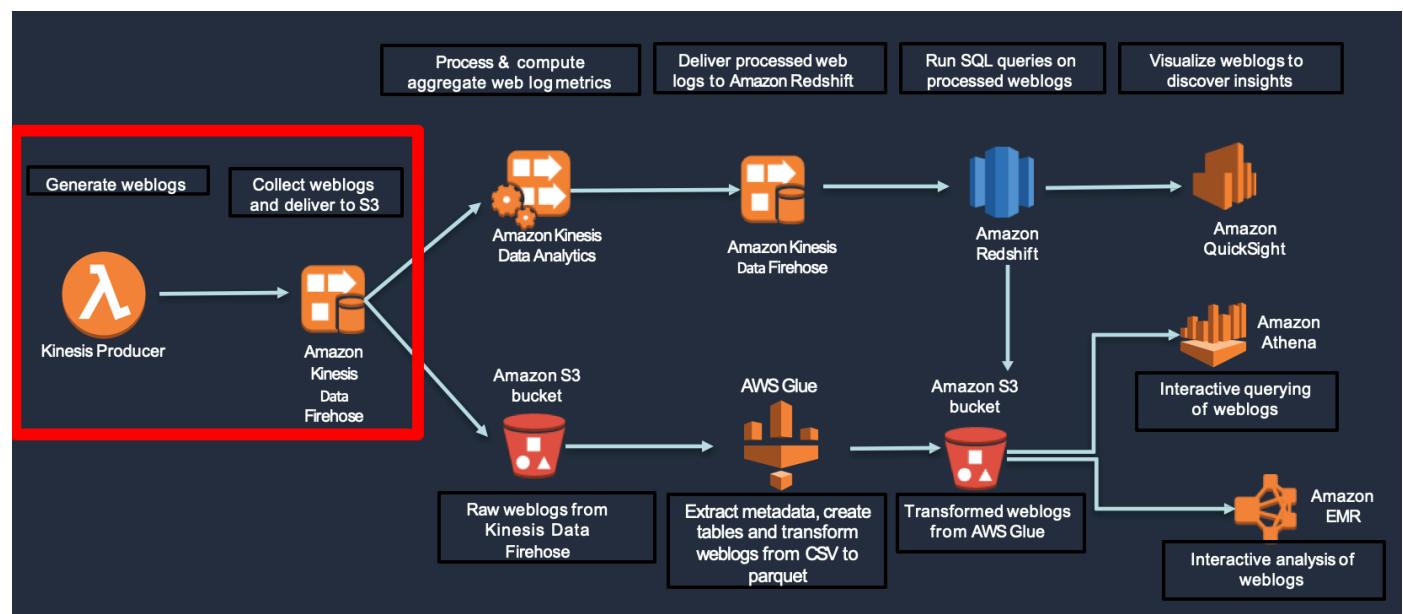
10. Click **Create Stack**.
  11. At this point, you will be directed back to the CloudFormation console and will see a status of **CREATE\_IN\_PROGRESS**. Do not continue until the status changes to **CREATE\_COMPLETE**.
- Note:** this step may take 15-20 min to complete.

Key	Value	Description
ApacheLogsFunction	bd-workshop-KinesisStack-1C-GenerateLogsLambdaFunc-1VMOU1GR4FBDT	Lambda function used to generate Apache Logs
EmrMasterNodeDns	ec2-34-205-39-95.compute-1.amazonaws.com	Public DNS name of the master EMR instance
ParquetLogsCrawler	ParquetLogsCrawler-9q3N1EryLEwL	Crawler in Glue to classify data from the output of converting raw data to parquet
RedshiftPGWeb	<a href="http://ec2-3-93-68-16.compute-1.amazonaws.com">http://ec2-3-93-68-16.compute-1.amazonaws.com</a>	Address for the PGWeb Instance
RedshiftSpectrumRole	bd-workshop-RedshiftStack-155-RedshiftSpectrumRole-1QVF6FB9O3D0AR	IAM Role for Redshift Spectrum
S3Bucket	bd-workshop-s3bucket-g4ghitpzjyf5	S3 Bucket that all data will be created under
SagemakerNotebookInstanceName	devNotebook	Name of the sagemaker notebook
SagemakerRoleArn	arn:aws:iam::413094830157:role/service-role/bd-workshop-SagemakerStack-1FDTV77EU-SagemakerRole-A0QJBZAYR9XU	IAM role name

# Task 1: Collect logs w/a Kinesis Data Firehose delivery stream (5-min)

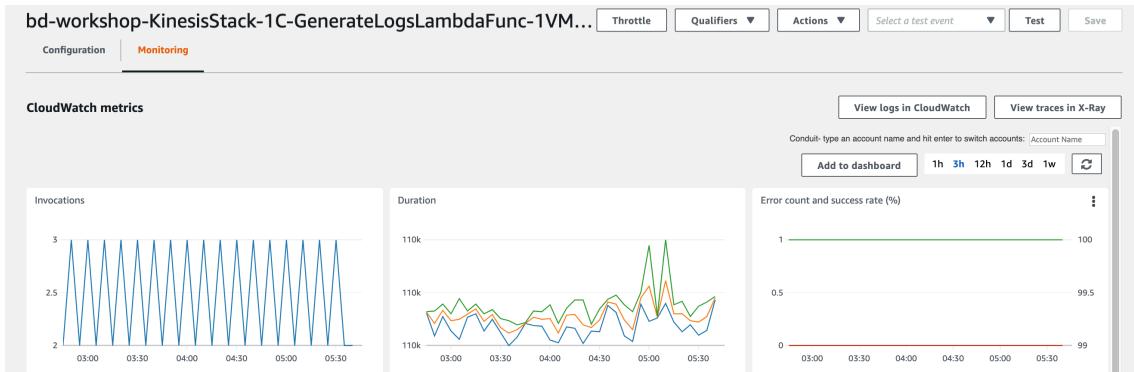
In this step, we are going to:

- Write to a Data Firehose delivery stream—Simulate writing transformed Apache weblogs to a Data Firehose delivery stream that is configured to deliver data into an S3 bucket.
- There are **many** different tools and libraries that can be used to write data to a Data Firehose delivery stream. One popular option is called the Amazon Kinesis Agent
- So that we don't have to install or set up software on your machine, we are going to use a Lambda function to simulate using the Amazon Kinesis Agent. The Lambda function can populate a Data Firehose delivery stream using a template and is simple to setup.



## 1.1 Verify Lambda function delivering logs

- 1.1.1 Go to the Lambda console and find a function named like:  
<StackName>-KinesisStack-xxx-GenerateLogsLambdaFunc-xxxxxx
- 1.1.2 Go to the Monitoring tab



### 1.1.3 Click the "View logs in CloudWatch" button

1.1.4 View the top log stream and you should see Apache log entries like below:

Filter events		all 2018-09-24 (16:10:42) ▾
Time (UTC +00:00)	Message	
2018-09-25		
▶ 16:02:43	START RequestId: 6f0337e3-c0dc-11e8-9f7a-71a603fe81fd Version: \$LATEST	
▶ 16:02:43	169.243.191.25 - - [25/Sep/2018:16:02:43 +0000] "POST /search/tag/list HTTP/1.0" 404 4989 "-" "Mozilla/5.0 (X11; Linux i686) AppleWebKit/5362 (KHTML, like Gecko) Chrome/15.0.826.0 Safari/53	
▶ 16:02:43	192.52.244.34 - - [25/Sep/2018:16:02:43 +0000] "GET /wp-admin HTTP/1.0" 200 5026 "-" "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_12_0 rv:3.0; tt-RU) AppleWebKit/534.15.5 (KHTML, like Ge	
▶ 16:02:43	198.51.98.170 - - [25/Sep/2018:16:02:43 +0000] "POST /wp-content HTTP/1.0" 200 5004 "-" "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 4.0; Trident/5.1)"	
▶ 16:02:43	192.175.63.43 - - [25/Sep/2018:16:02:43 +0000] "DELETE /list HTTP/1.0" 301 4960 "-" "Opera/8.33.(X11; Linux x86_64; hu-HU) Presto/2.9.181 Version/10.00"	
▶ 16:02:43	192.0.50.1 - - [25/Sep/2018:16:02:43 +0000] "POST /apps/cart.jsp?appId=8491 HTTP/1.0" 500 5023 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.2) AppleWebKit/532.28.1 (KHTML, like Gecko) Ver	
▶ 16:02:43	198.42.197.105 - - [25/Sep/2018:16:02:43 +0000] "POST /explore HTTP/1.0" 200 4968 "-" "Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 5.2; Trident/3.0)"	
▶ 16:02:43	169.168.111.2 - - [25/Sep/2018:16:02:43 +0000] "POST /search/tag/list HTTP/1.0" 200 4957 "-" "Mozilla/5.0 (Windows NT 6.1; ms-MY; rv:1.9.2.20) Gecko/2010-03-09 23:35:22 Firefox/3.8"	

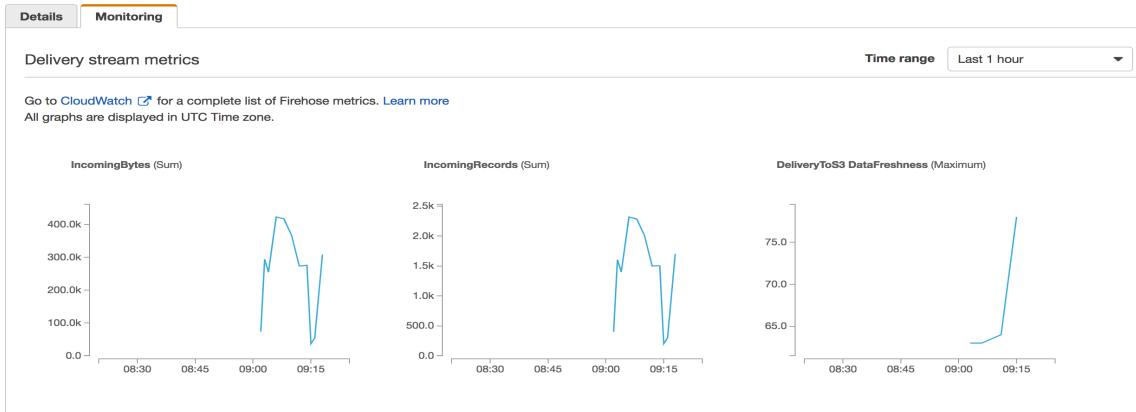
## 1.2 Monitoring Kinesis Data Firehose delivery to Amazon S3

1.2.1 Go to the [Kinesis console](#), click on “Data Firehose” and find data stream named like: <StackName>-KinesisStack-xxx-FirehoseDeliveryStream-xxxxxx

1.2.2 Select <StackName>-KinesisStack-xxx-FirehoseDeliveryStream-xxxxxx

Go to the **Monitoring** tab.

There should be metrics for delivery to Amazon S3. This might take some time to show up.



### 1.2.3 Go to Details tab and you can see the S3 bucket the Apache logs are being streamed to:

The screenshot shows the Amazon Kinesis Data Firehose Details page for a stream named `bd-workshop-KinesisStack-1C-FirehoseDeliveryStream-1TYD4G39JWI1C`. The stream ARN is `arn:aws:firehose:us-east-1:413094830157:deliverystream/bd-workshop-KinesisStack-1C-FirehoseDeliveryStream-1TYD4G39JWI1C`. The status is Active, created on 2019-07-19T23:27:0700, and the IAM role is `bd-workshop-KinesisStack-1C2IWUSWPVTVSF-S3Role-1R8ORQTBYB850`.

**Amazon S3 destination**

**S3 bucket:** `bd-workshop-s3bucket-g4ghitpzjyf5`

**Prefix:** `weblogs/raw/`

**Error prefix:** no error prefix specified

**Buffer conditions:** 50 MB or 60 seconds

**Compression:** GZIP

**Encryption:** Disabled

### 1.2.4 Click on S3 bucket and you can see that Apache logs are being streamed to this bucket:

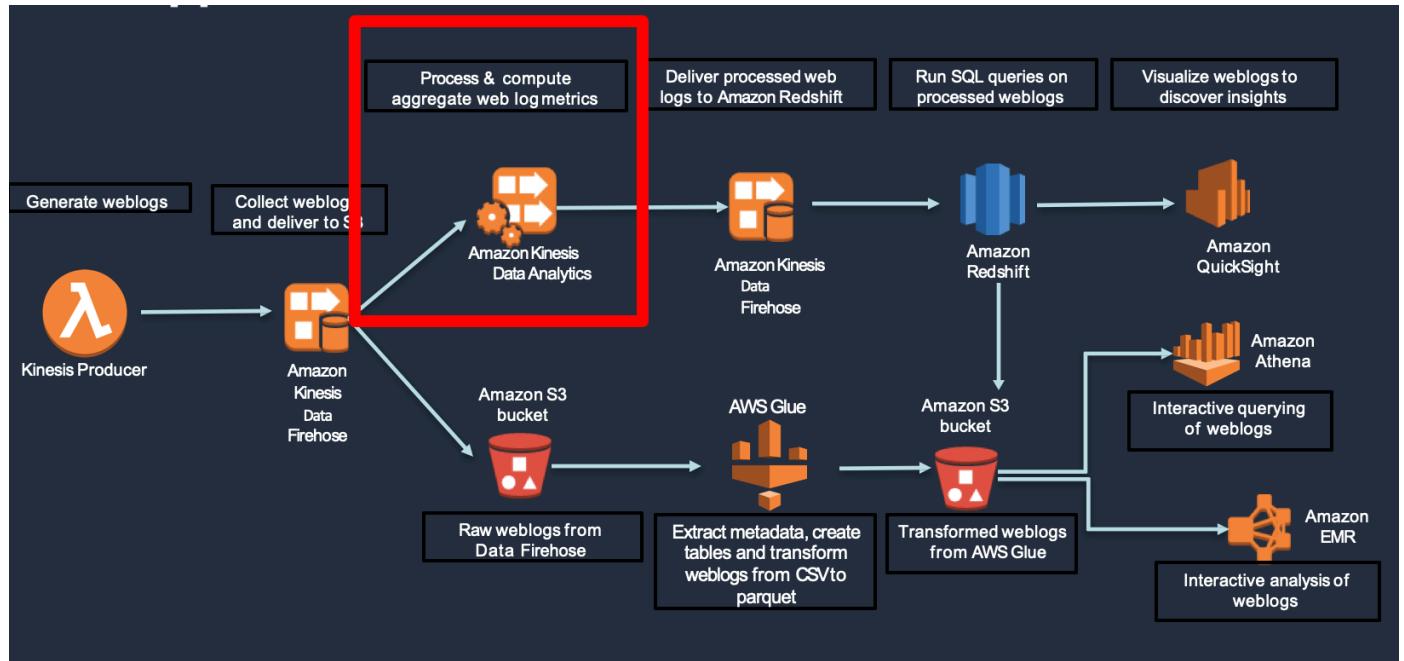
The screenshot shows the Amazon S3 Bucket Overview page for the bucket `bd-workshop-s3bucket-g4ghitpzjyf5`. The bucket contains 29 objects, all of which are files named after Kinesis Firehose delivery stream log entries. The objects were uploaded on July 19, 2019, at various times between 11:31:32 PM and 11:36:42 PM UTC. The storage class is Standard for all objects.

Name	Last modified	Size	Storage class
<code>bd-workshop-KinesisStack-1C-FirehoseDeliveryStream-1TYD4G39JWI1C-1-2019-07-20-06-30-29-4d1...</code>	Jul 19, 2019 11:31:32 PM GMT-0700	41.7 KB	Standard
<code>bd-workshop-KinesisStack-1C-FirehoseDeliveryStream-1TYD4G39JWI1C-1-2019-07-20-06-31-32-2838...</code>	Jul 19, 2019 11:32:35 PM GMT-0700	33.7 KB	Standard
<code>bd-workshop-KinesisStack-1C-FirehoseDeliveryStream-1TYD4G39JWI1C-1-2019-07-20-06-32-33-3028...</code>	Jul 19, 2019 11:33:36 PM GMT-0700	41.7 KB	Standard
<code>bd-workshop-KinesisStack-1C-FirehoseDeliveryStream-1TYD4G39JWI1C-1-2019-07-20-06-33-36-9b...</code>	Jul 19, 2019 11:34:39 PM GMT-0700	33.7 KB	Standard
<code>bd-workshop-KinesisStack-1C-FirehoseDeliveryStream-1TYD4G39JWI1C-1-2019-07-20-06-34-37-c428...</code>	Jul 19, 2019 11:35:40 PM GMT-0700	41.7 KB	Standard
<code>bd-workshop-KinesisStack-1C-FirehoseDeliveryStream-1TYD4G39JWI1C-1-2019-07-20-06-35-40-c9a4...</code>	Jul 19, 2019 11:36:42 PM GMT-0700	33.6 KB	Standard

## Task 2: Real-time data processing using Amazon Kinesis Data Analytics (20 min)

We are going to:

- Write a SQL query to compute an aggregate metric for an interesting statistic on the incoming data
- Write a SQL query using an anomaly detection function



### 2.1 Start Amazon Kinesis Data Analytics app

- Navigate to the [Kinesis console](#),
- Click on the Dashboard and then **Kinesis Analytics application**

**Amazon Kinesis dashboard**

Amazon Kinesis makes it easy to collect, process, and analyze video and data information. [What is streaming data?](#)

Kinesis data streams (1)	
Name	Status
weather_datastream	Active

[View all](#) [Create data stream](#)

Kinesis analytics applications (1)	
Name	Status
bd-workshop-KinesisStack-1C2IWUSWPTV...	Ready

[View all](#) [Create analytics application](#)

- Click on <stack-name>-KinesisStack-xxxxx-KinesisApplication

**bd-workshop-KinesisStack-1C2IWUSWPTVSF-KinesisApplication**

Application status: READY

Description: Kinesis Analytics Application Example

Application ARN: arn:aws:kinesisanalytics:us-east-1:413094830157:application/bd-workshop-KinesisStack-1C2IWUSWPTVSF-KinesisApplication

Application version ID: 2

**Source**

Streaming data

Connect to an existing Kinesis stream or Firehose delivery stream, or easily create and connect to a new demo Kinesis stream. Each application can connect to one streaming data source. [Learn more](#)

	Source	In-application stream name	ID	Record pre-processing
	Firehose delivery stream bd-workshop-KinesisStack-1C-FirehoseDeliveryStream-1TYD4G39JW11C	SOURCE_SQL_STREAM_001	1.1	bd-workshop-KinesisStack-1C2-PreProcessLambdaFunc-1LMF5UIG3WZAE

**Reference data (optional)**

Enrich data from your streaming data source with JSON or CSV data stored as an object in Amazon S3. Each application can connect to one reference data source. [Learn more](#)

[Connect reference data](#)

**Real time analytics**

Continuously analyze your source data with SQL. [Learn more](#)

[Go to SQL editor](#)

- Click on “Go to SQL editor”
- On the next screen, click on “Yes, start application”



The SQL editor is much more powerful when your application is running.

- See samples from your source data stream
- Get feedback on any errors in your configuration or SQL
- Watch as your data is processed in real-time by your SQL code



- View sample records in Kinesis app
- Review sample records delivered to the source stream (SOURCE\_SQL\_STREAM\_001)

ROWTIME	request_method	request_time	response_size	user_agent
2019-07-22 06:07:41.053	GET	2019-07-22 06:07:35.0	4987	Mozilla/5.0 (X11;
2019-07-22 06:07:41.053	POST	2019-07-22 06:07:35.0	5033	Mozilla/5.0 (Winc
2019-07-22 06:07:41.053	POST	2019-07-22 06:07:35.0	4966	Mozilla/5.0 (Winc
2019-07-22 06:07:41.053	GET	2019-07-22 06:07:35.0	5026	Opera/9.43.(Winc
2019-07-22 06:07:41.08	GET	2019-07-22 06:07:35.0	4973	Mozilla/5.0 (Maci

## 2.1 Calculate an aggregate metric

- Open the **KinesisAnalyticsSQL.txt** file located in the Scripts folder you downloaded at the beginning of the workshop. ([Workshop Scripts](#))
- Copy and paste the entire SQL of **KinesisAnalyticsSQL.txt** into the **SQL editor** of your Kinesis Data Analytics application, OVERWRITING the existing text
- Click “**Save and run SQL**”

```
48 /* Compute an anomaly score for each record in the input stream */
49 /* using Random Cut Forest */
50 /* Step 2 - Compute anomaly score */
51 CREATE OR REPLACE PUMP anomaly_pump AS INSERT INTO anomaly_stream
52 SELECT STREAM ROWTIME,
53     "host_address", "request_time", "request_method", "request_path", "request_protocol",
54     "response_code", "response_size", "referrer_host", "user_agent",
55     anomaly_score
56 FROM TABLE(RANDOM_CUT_FOREST(
57     CURSOR(SELECT STREAM * FROM source_table stream 001)));
58
```

Saving SQL  
Running SQL

- This will run for few seconds and will produce results.

Application status: RUNNING

ROWTIME	RESPONSE_CODE	REQUEST_TIME	REQUEST_COUNT	Avg Response
2019-07-22 06:44:03.045	500	2019-07-22 06:43:00.0	327	4997

## 2.2 Anomaly detection

Take a look at the anomaly detection section in the SQL script in the SQL Editor:

- It creates an **ANOMALY\_STREAM** with the attribute **anomaly\_score**
- It calculates the anomaly score for each record in the stream by using the built-in random cut forest function

Amazon Kinesis

- Dashboard
- Data Streams
- Data Firehose
- Data Analytics**
- Video Streams
- External resources
- What's new

## Real-time analytics

Save and run SQL Add SQL from templates Download SQL SQL reference guide

Kinesis data generator tool

```

32  /* Activity 2C: Anomaly detection */
33  /* Step 1 - Creates a stream for anomaly scores */
34  CREATE OR REPLACE STREAM anomaly_stream (
35    process_time      TIMESTAMP,
36    host_address      VARCHAR(512),
37    request_time      TIMESTAMP,
38    request_method    VARCHAR(5),
39    request_path      VARCHAR(1024),
40    request_protocol  VARCHAR(10),
41    request_query     VARCHAR(1024),
42

```

Application status: RUNNING

Source data Real-time analytics Destination

In-application streams:

- AGGREGATE\_STREAM
- ANOMALY\_STREAM
- 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	ANOMALY_SCORE
appleWebKit/531.35.3 (KHTML, like Gecko) Version/4.0.5 Mobile/8B115 Safari/6531.35.3	0.76
eko) Chrome/52.0.890.0 Safari/5312	0.81
eko) Chrome/52.0.887.0 Safari/5360	1.02
00	0.82

## 2.3 Kinesis Data Analytics in-application streams

In-application streams:

- Aggregate stream
- Anomaly stream
- Destination SQL stream
- Error stream

Amazon Kinesis

- Dashboard
- Data Streams
- Data Firehose
- Data Analytics**
- Video Streams
- External resources
- What's new

## Real-time analytics

Save and run SQL Add SQL from templates Download SQL SQL reference guide

Kinesis data generator tool

```

32  /* Activity 2C: Anomaly detection */
33  /* Step 1 - Creates a stream for anomaly scores */
34  CREATE OR REPLACE STREAM anomaly_stream (
35    process_time      TIMESTAMP,
36    host_address      VARCHAR(512),
37    request_time      TIMESTAMP,
38    request_method    VARCHAR(5),
39    request_path      VARCHAR(1024),
40    request_protocol  VARCHAR(10),
41    request_query     VARCHAR(1024),
42

```

Application status: RUNNING

Source data Real-time analytics Destination

In-application streams:

- AGGREGATE\_STREAM
- ANOMALY\_STREAM
- DESTINATION\_SQL\_STREAM
- error\_stream

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

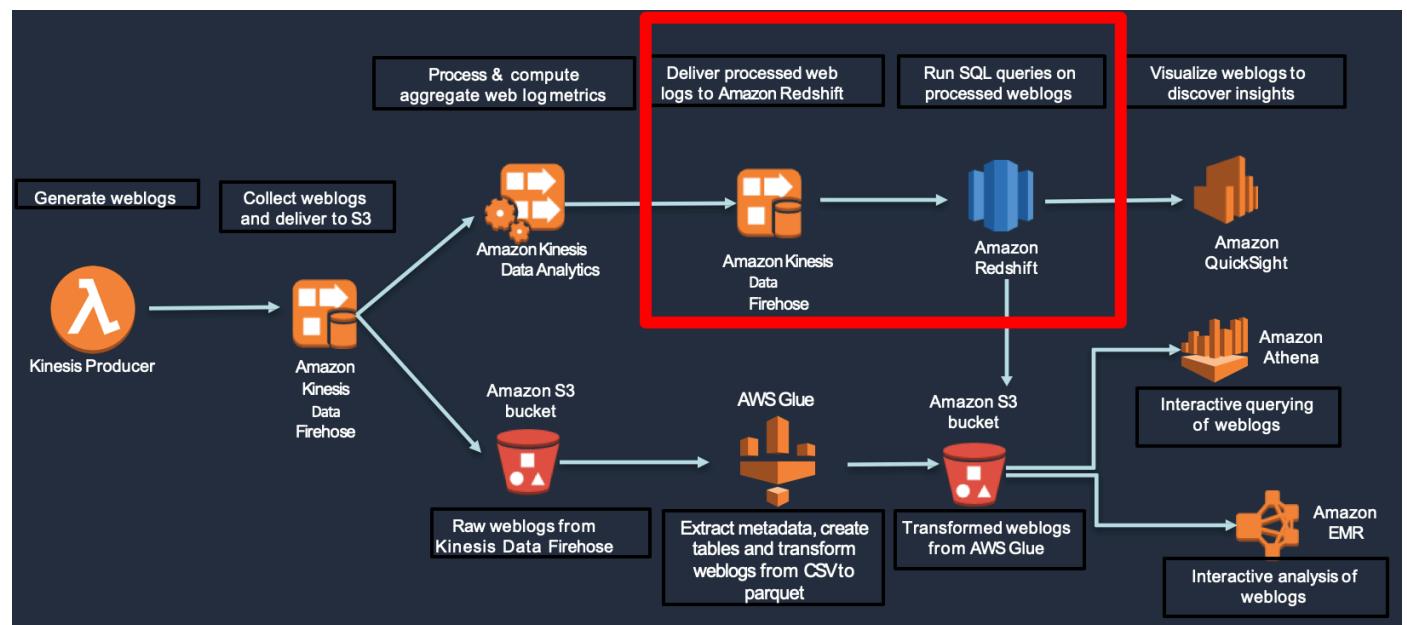
Scroll to bottom when new results arrive.

ROWTIME	process_time	host_address	request_time	request_method
2019-07-22 06:51:20.155	2019-07-22 06:51:20.155	198.253.108.29	2019-07-22 06:51:14.0	POST
2019-07-22 06:51:20.155	2019-07-22 06:51:20.155	198.56.45.142	2019-07-22 06:51:14.0	DELETE
2019-07-22 06:51:20.155	2019-07-22 06:51:20.155	203.0.112.167	2019-07-22 06:51:14.0	PUT

## Task 3: Deliver streaming results to Amazon Redshift using Kinesis Data Firehose (5 min)

We are going to:

- Connect to Amazon Redshift cluster and create a table to hold web logs data
- Update Kinesis Data Analytics application to send data to Amazon Redshift, via the Data Firehose delivery stream



### 3.1 Connect to Amazon Redshift Cluster

You can connect to Amazon Redshift cluster using one of below option

- PGWeb (already installed when you run CloudFormation template at the beginning)
- Use any JDBC/ODBC client: SQL Workbench/J or Aginity or DBeaver or Datagrip
- Use Redshift Query Editor (*This will not work for this workshop because we have created Redshift cluster of node type ds2.xlarge – which is currently unsupported by Redshift Query Editor*)

**Instructions for PGWeb:** Here are the steps when you want to use PGWeb to connect Amazon Redshift cluster created in the workshop.

To navigate to pgweb:

- Navigate to EC2 dasboard
- Click on <https://console.aws.amazon.com/ec2/>
- Click on Instances from left side panel

The screenshot shows the AWS EC2 Instances dashboard. At the top, there are buttons for 'Launch Instance', 'Connect', and 'Actions'. Below this is a search bar with 'pgweb' entered. A table lists one instance:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs
PGWeb	i-00fe08a4f0cb81ab0	m4.large	us-east-1c	running	2/2 checks ...	None	ec2-3-93-68-16.compute-1.amazonaws.com	3.93.68.16	-

Below the table, the instance details are shown: Instance: i-00fe08a4f0cb81ab0 (PGWeb) Public DNS: ec2-3-93-68-16.compute-1.amazonaws.com. There are tabs for Description, Status Checks, Monitoring, and Tags. The Status Checks tab is selected. At the bottom, it shows Instance ID: i-00fe08a4f0cb81ab0.

- Select pgweb instance and copy the **Public DNS (IPv4)**
- Paste in your browser and Start interacting with pgweb

**Instructions for SQL Workbench/J:** Detailed instructions to set up SQL Workbench/J are [here](#).

In short, follow the steps below:

- Go to the SQL Workbench/J website [here](#).
- Download and install the Current stable version for your operating system.
- Download the Amazon Redshift JDBC Driver [here](#) and save it in your local system.

## 3.2 Create table in Amazon Redshift

- Make sure you are in the **logs** database
- Use **RedshiftSQL.txt** from Workshop Scripts for the CREATE TABLE statement
- Copy CREATE TABLE SQL statement and paste in the editor and run query to create the Redshift table '**weblogs**'.
- Table **weblogs** will capture incoming data from Kinesis Data Firehose delivery stream.

```
CREATE TABLE weblogs
(
    row_time timestamp encode raw,
    host_address varchar(512) encode lzo,
    request_time timestamp encode raw,
    request_method varchar(5) encode lzo,
    request_path varchar(1024) encode lzo,
    request_protocol varchar(10) encode lzo,
    response_code int encode delta,
    response_size int encode delta,
    referrer_host varchar(1024) encode lzo,
    user_agent varchar(512) encode lzo
) DISTSTYLE EVEN
```

```
SORTKEY (request_time);
```

### 3.3 Deliver data to Amazon Redshift using Data Firehose

Update Kinesis Data Analytics application to send data to Kinesis Data Firehose delivery stream. Data Firehose delivers the streaming data to Amazon Redshift.

#### 3.3.1 Go to the [Kinesis Data Analytics console](#).

#### 3.3.2 Select the KinesisApplication

The screenshot shows the Kinesis Analytics applications console. On the left, there's a sidebar with 'Amazon Kinesis' at the top, followed by 'Dashboard', 'Data Streams', 'Data Firehose', 'Data Analytics' (which is selected and highlighted in orange), and 'Video Streams'. Below that are 'External resources' and 'What's new'. The main area is titled 'Kinesis Analytics applications' and contains a sub-header: 'Kinesis Analytics applications continuously read and process data from streaming sources in real-time. [Learn more](#)'. There are 'Create application' and 'Actions' buttons. A search bar says 'Filter or search by application name'. Below that is a table with one row. The table has columns for 'Application name', 'Runtime', and 'State'. The application name is 'bd-workshop-KinesisStack-1C2IWUSWPTVSF-KinesisApplication', runtime is 'SQL', and state is 'Running'. To the right of the table are 'Created' (Jul 19, 2019 11:28:27 PM) and 'Last updated' (Jul 21, 2019 11:40:57 PM). A blue button labeled 'Application details' is at the bottom right of the table.

#### 3.3.3 Click on Application Details and scroll down until you see "Destination".

The screenshot shows the 'Destination' section of the Kinesis Analytics application details. It features a circular icon with three downward arrows. The section title is 'Destination' with a subtitle '(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.' Below this are 'Connect new destination' and 'Disconnect destination' buttons. A table lists the destination. The table has columns for 'Destination', 'In-application stream name', and 'ID'. One row is shown: 'Firehose delivery stream bd-workshop-RedshiftStack-1-RedshiftDeliveryStream-YDSY9379ZDIH' with 'DESTINATION\_SQL\_STREAM' as the in-application stream name and '2.1' as the ID.

#### 3.3.4 Choose the Amazon Redshift delivery stream as destination and click on the edit button (see the pencil icon in the figure below).

#### 3.3.5 Validate your destination

- 3.3.5.1 Validate that the Kinesis Data Firehose stream is "<stackName>RedshiftStack-xxx-**RedshiftDeliveryStream**-xxxxxx"
- 3.3.5.2 Keep the default for "Choose an existing in-application stream".

## DESTINATION\_SQL\_STREAM

- 3.3.5.3 Make sure CSV is the “Output format”
- 3.3.5.4 Validate that “Choose from IAM roles that Kinesis Data Analytics can assume”
- 3.3.5.5 Click “Save and continue”

The screenshot shows the 'Edit destination connection' page in the Amazon Kinesis Data Analytics console. The left sidebar shows 'Amazon Kinesis' with 'Data Streams' and 'Data Firehose' collapsed, and 'Data Analytics' selected. Below that are 'Video Streams', 'External resources', and 'What's new'. The main area has a title 'Edit destination connection' with a note: 'You are editing the destination connection to Firehose delivery stream bd-workshop-RedshiftStack-1-RedshiftDeliveryStream-YDSY9379ZDIH, which is connected to the in-application stream name DESTINATION\_SQL\_STREAM.' A 'Destination\*' section has three options: 'Kinesis stream' (radio button), 'Kinesis Firehose delivery stream' (radio button, selected), and 'AWS Lambda function' (radio button). Below it is a dropdown for 'Kinesis Firehose delivery stream\*' containing 'bd-workshop-RedshiftStack-1-RedshiftDeliveryStream-YDSY9379ZDIH' with a 'View' link. A 'Create new' button is also present. The 'In-application stream' section says 'In-application streams are continuous flows of data records. You create in-application streams in SQL to contain the data you want to persist to the specified destination. [Learn more](#)'. It has a 'Connect in-application stream' section with two options: 'Choose an existing in-application stream' (radio button, selected) and 'Specify a new in-application stream name' (radio button). The 'In-application stream name\*' dropdown contains 'DESTINATION\_SQL\_STREAM'. The 'Output format' section has 'JSON' and 'CSV' radio buttons, with 'CSV' selected. The 'Access permissions' section says 'Create or choose IAM role with the required permissions. [Learn more](#)'. It has an 'Access permissions\*' dropdown with two options: 'Create / update IAM role kinesis-analytics-bd-workshop-KinesisStack-1C2IWUSWPTV-us-east-1' (radio button) and 'Choose from IAM roles that Kinesis Analytics can assume' (radio button, selected). The 'IAM role\*' dropdown contains 'bd-workshop-KinesisStack-1C2I-KinesisAnalyticsRole-1QZ1D84GS1902' with a 'View' link. A note at the bottom says 'Only IAM roles with the [required trust policy](#) attached are available for selection.'

Note: It will take about 1–2 minutes for everything to be updated and for data to start appearing into Amazon Redshift.

## 3.4 Amazon Redshift - test queries

- Find distribution of response codes over days (copy SQL from RedshiftSQL file)

```
SELECT TRUNC(request_time), response_code, COUNT(1)
FROM weblogs
GROUP BY 1,2
```

```
ORDER BY 1,3 DESC;
```

- Count the number of 404 response codes

```
SELECT COUNT(1)
FROM weblogs
WHERE response_code = 404;
```

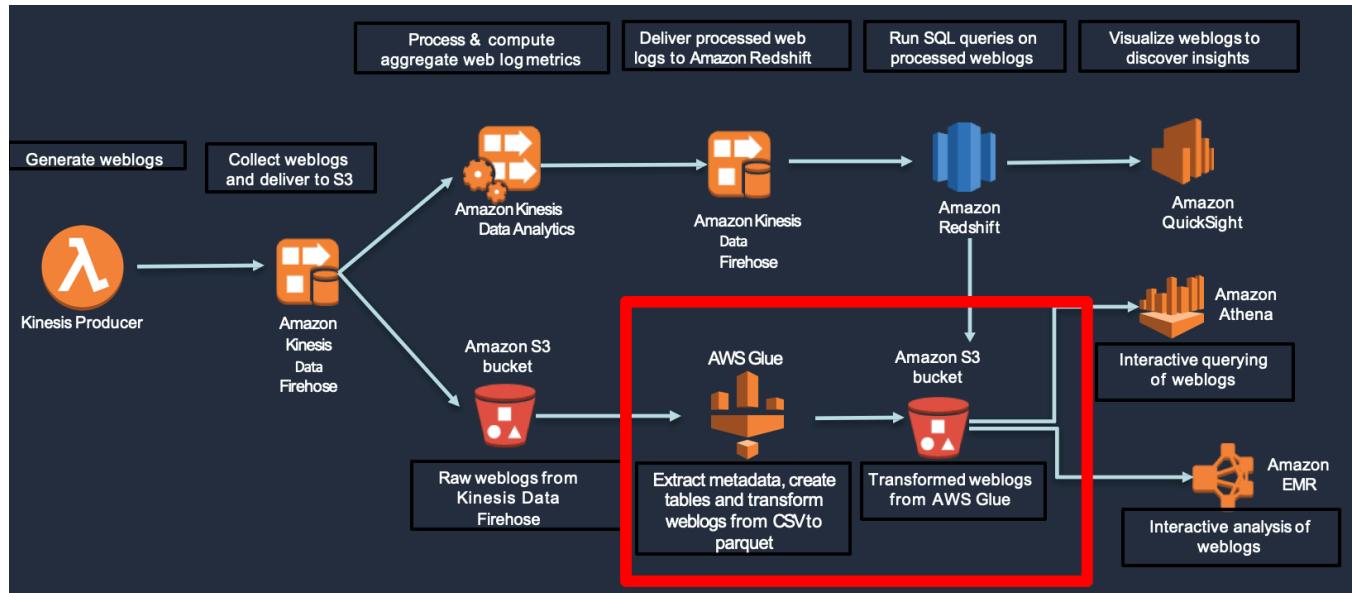
- Show all requests paths with status "PAGE NOT FOUND"

```
SELECT TOP 1 request_path, COUNT(1)
FROM weblogs
WHERE response_code = 404
GROUP BY 1
ORDER BY 2 DESC;
```

## Task 4: Transform weblogs to parquet format using AWS Glue (30 min)

We are going to:

1. Discover and catalog the weblog data deposited into the Amazon S3 bucket using AWS Glue crawler
2. Transform weblogs to parquet format using the AWS Glue ETL job authoring tool



### 4.1 Discover dataset with AWS Glue

We will use AWS GLUE's crawler to extract data and metadata.

- From the AWS Management Console,
- Select / search **AWS Glue**.
- Click on “Get Started” on the next screen (if appears)
- Select "**Crawlers**" section on the left-hand side panel
- Click on "**Add crawler**"

Screenshot of the AWS Glue Crawler interface. The sidebar shows "Crawlers" selected. The main area displays a table with columns: Name, Schedule, Status, Logs, Last runtime, Median runtime, Tables updated, and Tables added. A message at the bottom states "You don't have any crawlers yet." with a "Add crawler" button.

- Specify a name for the crawler. Click "Next"

**Add crawler**

Add information about your crawler

Crawler info

Crawler source type

Data store

IAM Role

Schedule

Output

Review all steps

Crawler name

Tags, description, security configuration, and classifiers (optional)

Catalog options (optional)

Next

- Select Crawler source type to "Data Stores" and click "Next"

**Specify crawler source type**

Choose Existing catalog tables to specify catalog tables as the crawler source. The selected tables specify the data stores to crawl. This option doesn't support JDBC data stores.

**Crawler source type**

Data stores

Existing catalog tables

Back      Next

- Provide S3 path location where the raw weblogs were placed (navigate to S3 path: s3://<S3 bucket from template>/weblogs/raw)

**Choose S3 path**

aws-glue-scripts-413094830157-us-east-2

aws-glue-temporary-413094830157-us-east-1

aws-logs-413094830157-us-east-1

aws-logs-413094830157-us-west-2

awpsa-aod-recordings-2018

awpsa-redshift-cloudtrail-bucket

awpsa-redshift-guarddutylogs

awpsa-redshift-lab

awpsa-sampleddata

bd-workshop-s3bucket-g4ghitpzjif5

emrlogs

weblogs

processed

raw

cf-templates-10521p1nfd0yo-eu-west-1

Select

## Add a data store

**Choose a data store**

S3

**Crawl data in**

Specified path

**Include path**

s3://bd-workshop-s3bucket-g4ghitpzyjf5/weblogs/raw 

All folders and files contained in the include path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

► Exclude patterns (optional)

[Back](#) [Next](#)

- Click "Next"
- Add Another data store – choose No.

## Add another data store

Yes  
 No

[Back](#) [Next](#)

- Click "Next"
- In the IAM role section, select “Choose an existing IAM role”
- Select role <StackName>-GlueStack-xxx-GlueCrawlerRole-xxxxxx as the IAM role

### Choose an IAM role

The IAM role allows the crawler to run and access your Amazon S3 data stores. [Learn more](#)

Update a policy in an IAM role  
 Choose an existing IAM role  
 Create an IAM role

**IAM role** 

bd-workshop-GlueStack-14IJHX0DS5AX-GlueCrawlerRole-NVD353IIH751 

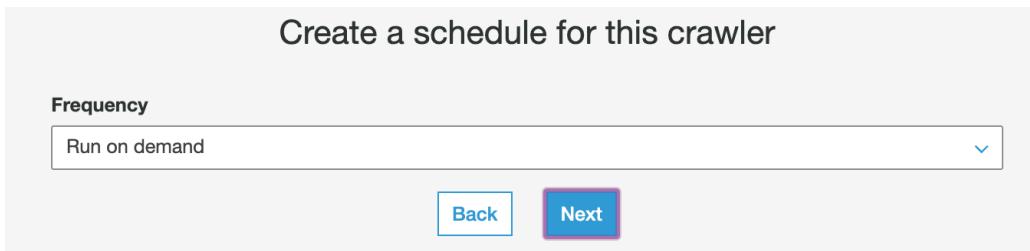
This role must provide permissions similar to the AWS managed policy, **AWSGlueServiceRole**, plus access to your data stores.

- s3://bd-workshop-s3bucket-g4ghitpzyjf5/weblogs/raw

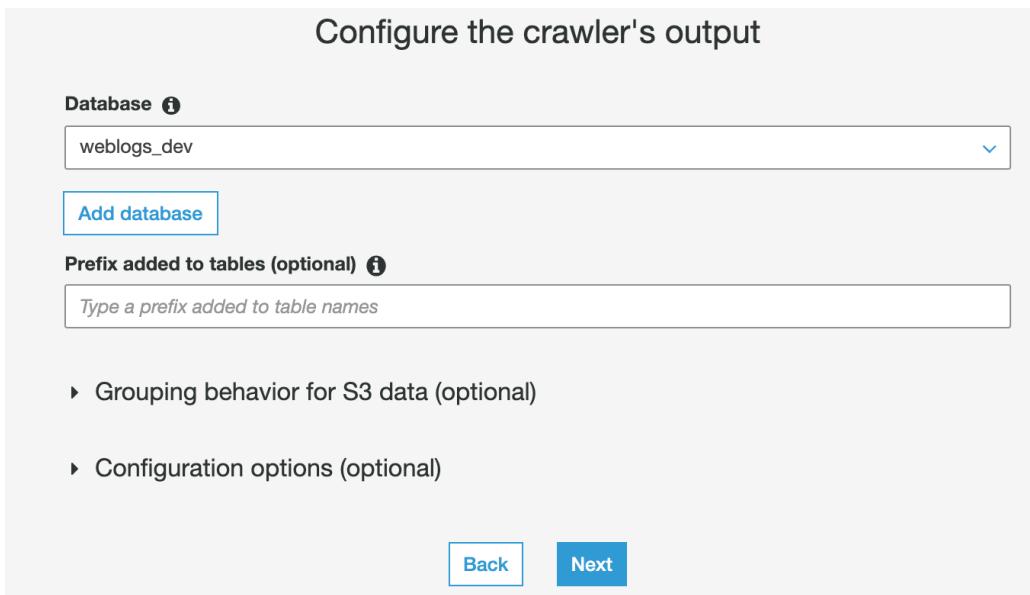
You can also create an IAM role on the [IAM console](#).

[Back](#) [Next](#)

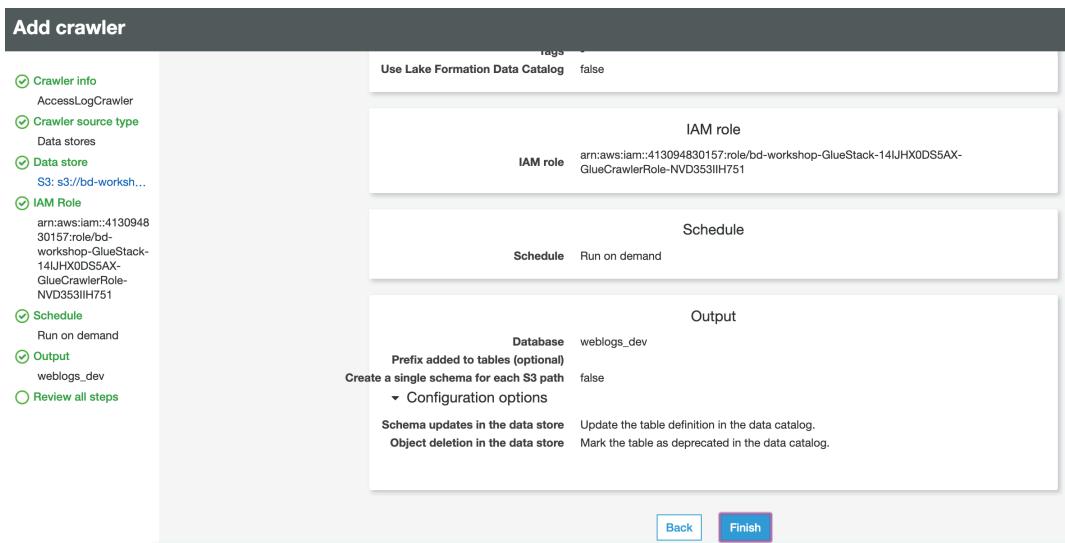
- Click "Next"
- Choose "Run on demand" to run the crawler now, and click "Next"



- On the next screen, drop down Database and select “**weblogs\_dev**” and Click “Next”



- Review and click "Finish" to create a crawler



- Click on "Run it now?" link to run the crawler

Crawler **AccessLogCrawler** was created to run on demand. [Run it now?](#)

<input type="checkbox"/>	Name	Schedule	Catalog type	Status	Logs	Last runtime
<input checked="" type="checkbox"/>	AccessLogCrawler		Glue	Ready	<a href="#">Logs</a>	0 s

- Now wait for 1-2 min and Crawler shows a **Ready** status when it is finished running

Glue

Data catalog

Databases

Tables

Connections

**Crawlers**

Classifiers

Settings

Crawler "AccessLogCrawler" completed and made the following changes: 1 tables created, 0 tables updated. See the tables created in database [weblogs\\_dev](#).

Name	Schedule	Catalog type	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
AccessLogCrawler		Glue	Ready	<a href="#">Logs</a>	1 min	1 min	0	1

- Observe that the crawler has created a table for your dataset
- The crawler automatically classified the dataset as **combinedapache** log format

AWS Glue

Data catalog

Databases

**Tables**

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
raw	weblogs_dev	s3://bd-workshop-s3bucket-g4ghitpzjyjf5/weblogs/raw/	combinedapache

- Click the table to take a look at the properties

Tables > raw

Last updated 22 Jul 2019 Table Version (Current version) ▾

[Edit table](#) [Delete table](#) [View partitions](#) [Compare versions](#) [Edit schema](#)

Name	raw
Description	
Database	weblogs_dev
Classification	combinedapache
Location	s3://bd-workshop-s3bucket-g4ghitpzjyjf5/weblogs/raw/
Connection	
Deprecated	No
Last updated	Mon Jul 22 01:14:39 GMT-700 2019
Input format	org.apache.hadoop.mapred.TextInputFormat
Output format	org.apache.hadoop.hive.io.HiveIgnoreKeyTextOutputFormat
Serde serialization lib	com.amazonaws.glue.serde.GrokSerDe
Serde parameters	input.format %{COMBINEDAPACHELOG}
Table properties	sizeKey 111634315 objectCount 2849 UPDATED_BY_CRAWLER <b>AccessLogCrawler</b> CrawlerSchemaSerializerVersion 1.0 recordCount 46646 averageRecordSize 179 grokPattern %{COMBINEDAPACHELOG} CrawlerSchemaDeserializerVersion 1.0 compressionType gzip typeOfData file

Schema

Showing: 1 - 15 of 15 < >

Column name	Data type	Partition key	Comment
1 clientip	string		
2 ident	string		

- AWS Glue used the **GrokSerDe** (Serializer/Deserializer) to correctly interpret the weblogs
- You can click on the "**View partitions**" link to look at the partitions in the dataset

Tables > raw

Last updated 22 Jul 2019 Table Version (Current version)

Edit table Delete table Close partitions Compare versions Edit schema

Showing: 1 - 51 < >

partition_0	partition_1	partition_2	partition_3		
2019	07	20	12	<a href="#">View files</a>	<a href="#">View properties</a>
2019	07	21	03	<a href="#">View files</a>	<a href="#">View properties</a>
2019	07	21	11	<a href="#">View files</a>	<a href="#">View properties</a>
2019	07	20	14	<a href="#">View files</a>	<a href="#">View properties</a>
2019	07	21	06	<a href="#">View files</a>	<a href="#">View properties</a>
2019	07	21	15	<a href="#">View files</a>	<a href="#">View properties</a>
2019	07	20	06	<a href="#">View files</a>	<a href="#">View properties</a>

## 4.2 Create ETL job in AWS Glue

- With the dataset cataloged and table created, we are now ready to convert the weblogs-based Apache combined log format to a more optimal parquet format for querying.
- Click on "**Add job**" to begin creating the ETL job.

AWS Glue

Jobs A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers.

Add job Action Filter by attributes

Name	Script location	Last modified

You don't have any jobs defined yet.

Add job

Data catalog

Databases Tables Connections Crawlers Classifiers

ETL Jobs Triggers Dev endpoints

- Job name: **accesslogetl1**
- Select the IAM role <StackName>-GlueStack-xxx-GlueJobRole-xxxxxxxx
- Select Type as Spark
- Select "a new script to be authored by you" Script file name: **glue-workshop-etl.py**
- For the S3 path where the script will be stored, use path **s3://<S3 bucket from template>/script**
- For the Temporary Directory, use path **s3://<S3 bucket from template>/temp**

Configure the job properties

<b>Name</b>	accesslogetl1
<b>IAM role</b> ⓘ	bd-workshop-GlueStack-14IJHX0DS5AXB-GlueJobRole-FGPM74Z1M8M4
Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job. <a href="#">Create IAM role</a> .	
<b>Type</b>	Spark
<b>Glue version</b>	Spark 2.2, Python 2 (Glue version 0.9)
<b>This job runs</b>	
<input checked="" type="radio"/> A proposed script generated by AWS Glue ⓘ <input type="radio"/> An existing script that you provide <input type="radio"/> A new script to be authored by you	
<b>Script file name</b>	glue-workshop-etl.py
<b>S3 path where the script is stored</b>	s3://bd-workshop-s3bucket-g4ghitpzjyjf5/script
<b>Temporary directory</b> ⓘ	s3://bd-workshop-s3bucket-g4ghitpzjyjf5/temp/

- **DO NOT CLICK NEXT JUST YET**
- Expand **Security configuration, script libraries, and job parameters (optional)** section,
  - Set the Maximum capacity (DPUs) to 10.
  - Let's pass a job parameter to send the S3 path where parquet files will be deposited.
  - Specify the following values for Key and Value
  - **Key: --parquet\_path** (notice the 2 hyphens at the beginning and underscore between "parquet" and "path")
  - **Value: s3://<S3 bucket from template>/weblogs/parquet**

▼ Security configuration, script libraries, and job parameters (optional)

**Security configuration**

None

The security configuration specifies how the script is encrypted using server-side encryption with AWS KMS-managed keys (SSE-KMS) or Amazon S3-managed encryption keys (SSE-S3).

Server-side encryption

**Python library path**  
s3://bucket-name/folder-name/file-name

**Dependent jars path**  
s3://bucket-name/folder-name/file-name

**Referenced files path**  
s3://bucket-name/folder-name/file-name

**Worker type**  
Standard

**Maximum capacity**  
10

**Max concurrency**   
1

**Job timeout (minutes)**   
2880

**Delay notification threshold (minutes)**

**Number of retries**  
0

**Job parameters**

Key	Value
--parquet_path	zyjf5/weblogs/parquet
Type key...	Type value...

- Click "Next" in the following screen
- Choose any source path and Next
- Choose any target path and Next
- Review and click "Save Job" to create the job

Settings

ETL

Workflows

**Jobs**

ML  
Transforms

Triggers

Dev endpoints

Notebooks

Security

Security configurations

History Details Script Metrics

Name	convert-tpc-to-parquet	Python lib path	-
IAM role	AWS-Glue-ServiceRole	Jar lib path	-
Type	Spark	Other lib path	-
Python version	2	Parameters	-
Catalog type	Glue	Connections	-
ETL language	python	Maximum capacity	10
Script location	s3://aws-glue-scripts-413094830157-us-east-1/admin/convert-tpc-to-parquet	Job timeout (minutes)	2880
Temporary directory	s3://aws-glue-temporary-413094830157-us-east-1/admin	Delay notification threshold (minutes)	-
Job bookmark	Disable	Tags	-
Job metrics	Disable		
Continuous logging	Disable		
Server-side encryption	Disabled		

Automatically run this job if any of the following triggers fire:

- Select the ETL job “accesslogetl1” and click on “Edit Script”.

History Details Script Metrics

```

1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 from pyspark.context import SparkContext
9
10 ## @params: [JOB_NAME]
11 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
12
13 sc = SparkContext()
14 glueContext = GlueContext(sc)
15 spark = glueContext.spark_session
16 job = Job(glueContext)
17 job.init(args["JOB_NAME"], args)
18 ## @type: Dataframe

```

Edit script

- Copy the ETL code in `glue-workshop-etl.py` from `BigDataWorkshop.zip` and paste into ETL job editor. (Overwrite; don't append.)

The screenshot shows the AWS Glue ETL job editor interface. At the top, there are buttons for Action (dropdown), Save, Run job, Generate diagram, and Help. To the right, there is a link to "Insert template at cur". Below the buttons, a message says "The diagram cannot be generated. Check the annotations in your script." There are plus and minus icons for expanding/collapsing code sections. A help icon is also present. The main area contains the Python script code:

```

3  from awsglue.utils import getResolvedOptions
4  from pyspark.context import SparkContext
5  from awsglue.context import GlueContext
6  from awsglue.job import Job
7  from pyspark.sql.functions import *
8  from awsglue.dynamicframe import DynamicFrame
9
10
11 ## @params: [JOB_NAME]
12 args = getResolvedOptions(sys.argv, ['JOB_NAME', 'parquet_path'])
13
14 # Create Glue Context and spark session
15 sc = SparkContext()
16 glueContext = GlueContext(sc)
17 spark = glueContext.spark_session
18 job = Job(glueContext)
19 job.init(args['JOB_NAME'], args)
20
21 # Input: Database and table name from Catalog
22 db_name = "weblogs_dev"
23 table_name = "raw"
24
25

```

At the bottom, there are tabs for Logs and Schema, with Schema being selected.

- Ensure that the `db_name` and `table_name` statements reflect the database and table name created by the AWS Glue crawler.
- Click "**Save**" and "**Run job**" button to execute your ETL
- Make sure that the `--parquet_path` is correctly set in the job parameters.
- Check that the script name is `glue-workshop-etl.py`

## Parameters (optional)

Review and override parameter values, as needed, before running this job. Changes affect this run only. Edit a job to change default parameter values.

- ▶ Advanced properties
- ▶ Monitoring options
- ▶ Tags
- ▶ Security configuration, script libraries, and job parameters

Only job **accesslogetl1** is run. Jobs dependent on the completion of job **accesslogetl1** will not be run. To run a job and trigger dependent jobs, define an on-demand trigger.

**Run job**

**Parameters (optional)**

**Dependent jars path**  
s3://bucket-name/folder-name/file-name 

**Referenced files path**  
s3://bucket-name/folder-name/file-name 

**Worker type**  
Standard 

**Maximum capacity**  
10

**Job timeout (minutes)**  Sets timeout execution limit in minutes.

**Delay notification threshold (minutes)** 

**Job parameters**

Key	Value	X
--parquet_path	s3://bd-workshop-s3bucket-	
Type key...	Type value...	

**Run job**

- Click "Run job" to continue.

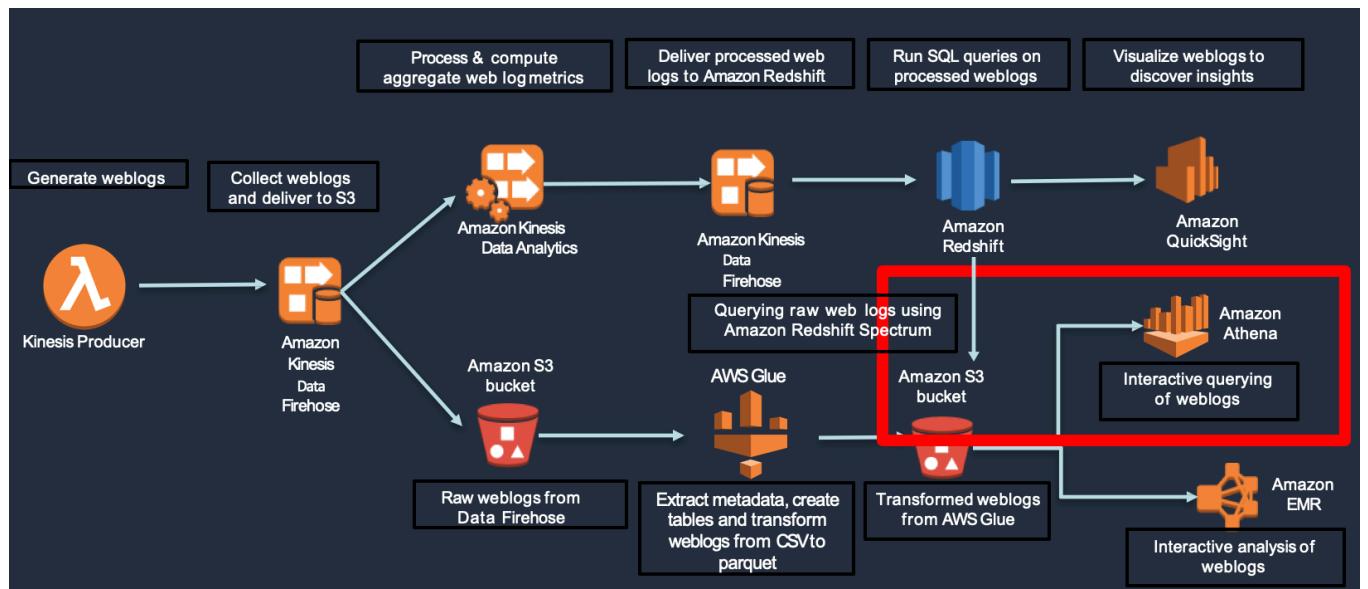
Note: This might take a few minutes. When the job finishes, weblogs will be transformed to parquet format.

- Go to s3://bd-workshop-s3bucket-xxxxxxxxxx/weblogs/parquet and verify that you see parquet.snappy files in there.

# Task 5: Querying Amazon S3 data using Redshift Spectrum and Amazon Athena (30 min)

We are going to:

- Create a table over the processed weblogs in Amazon S3 using an AWS Glue crawler. These are the parquet files created by AWS Glue ETL job in the previous section.
- Run queries from Amazon Redshift on the parquet weblogs in Amazon S3 using Amazon Redshift Spectrum.
- Run interactive queries from Athena on parquet weblogs in S3.



## 5.1 Set up AWS Glue crawler for processed parquet data

- Go to Crawlers in AWS Glue and search for **ParquetLogs**.
- Select the **ParquetLogsCrawler-xxxxxx**.

A screenshot of the AWS Glue Crawler list page. The sidebar shows 'Crawlers' is selected. The main table lists three crawlers:

Name	Schedule	Catalog type	Status	Logs	Last runtime	Median runtime
AccessLogCrawler		Glue	Ready	Logs	1 min	1 min
ParquetLogsCrawler-9q3N1EryLewL		Glue	Ready	Logs	0 secs	0 secs
ParquetLogsCrawler-9q3N1EryLewL		Glue	Ready	Logs	7 mins	7 mins

- Go to details tab and check that it's pointing to the right parquet location in Amazon S3.

[Run crawler](#)[Edit](#)

Name	ParquetLogsCrawler-9q3N1EryLEwL
Description	
Create a single schema for each S3 path	false
Security configuration	
Tags	-
State	Ready
Schedule	
Catalog type	Glue
Last updated	Fri Jul 19 23:22:24 GMT-700 2019
Date created	Fri Jul 19 23:22:24 GMT-700 2019
Database	weblogs_dev
Service role	bd-workshop-GlueStack-14IJHX0DS5AX-GlueCrawlerRole-NVD353IIH751
Selected classifiers	
Data store	S3
Include path	s3://bd-workshop-s3bucket-g4ghitpzjif5/weblogs/parquet
Exclude patterns	

## Configuration options

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

- Click “Run crawler” (Note - Should take about 1-2 min for the crawler to finish running)
- You will see new table named “parquet” in Glue Catalog.

Name	Database	Location	Classification	Last updated
parquet	weblogs_dev	s3://bd-workshop-s3bucket-g4ghitpzjif5/weblogs/parquet/	parquet	22 July 2019 2:28 AM UTC-7
raw	weblogs_dev	s3://bd-workshop-s3bucket-g4ghitpzjif5/weblogs/raw/	combinedapache	22 July 2019 1:14 AM UTC-7

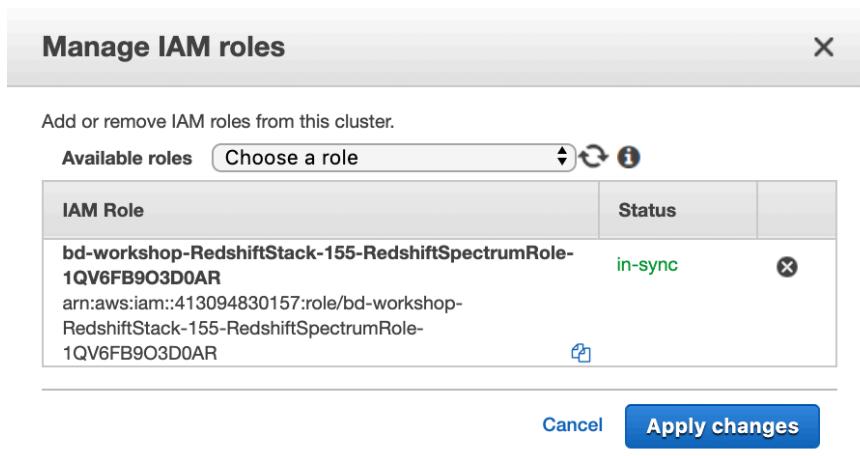
- Take a look at the schema of the parquet table

Column name	Data type	Partition key	Comment
1	string		
2	string		
3	string		
4	string		
5	string		
6	timestamp		
7	int		
8	int		

Note: Should have eight columns and Classification should be parquet

## 5.2 Query using Amazon Redshift Spectrum

- Navigate to Amazon Redshift-Clusters. Select the cluster with the <stack-name>
- Click on “Manage IAM roles”.
- Validate that IAM role shows  
bd-workshop-RedshiftStack-xxx-RedshiftSpectrumRole-xxxxxxxxxxxx.



- Navigate to pgWeb or SQLWorkbench.

Note: If you need the URL for pgWeb, check public IP of the PGWeb EC2 instance.

- Get the **RedshiftSpectrumSetup.sql** from the BigDataWorkshop.zip.
  - Make sure you **replace the IAM role ARN** with your SpectrumRole ARN.
  - Make sure **database name** is pointing to the **weblogs\_dev**.
  - This will create an external schema in Amazon Redshift called “Spectrum”.

```
create external schema spectrum
from data catalog
database 'weblogs_dev'
iam_role 'arn:aws:iam::413094830157:role/bd-workshop-RedshiftStack-155-RedshiftSpectrumRole-1QV6FB9O3D0AF'
create external database if not exists;
```

- Let's run a couple of simple queries against Amazon S3 from Amazon Redshift using Amazon Redshift Spectrum
- Count of all records in the parquet location in Amazon S3

```
Select count(*) from spectrum.parquet
```

- Count of all records in the parquet location in Amazon S3 where the response is 404

```
Select count(1) from spectrum.parquet where response=404;
```

## 5.3 Querying using Amazon Athena

- Athena allows you to run interactive queries against parquet data in Amazon S3
- Run the queries from the AthenaSQL from the BigDataWorkshop.zip
  - Count of each clientIP
  - Count of each response code

```
-- Access Log Queries
SELECT clientip, COUNT(1) as cnt FROM weblogs_dev.parquet GROUP BY 1;

-- Count of requests by response
SELECT response, COUNT(1) as cnt
FROM weblogs_dev.parquet
GROUP BY 1
ORDER BY 1,2 DESC;

-- Count of 200 response codes
SELECT COUNT(1) as cnt
FROM weblogs_dev.parquet
WHERE response = 200;

-- Get top request path for 301 response codes
SELECT request, COUNT(1)
FROM weblogs_dev.parquet
WHERE response = 301
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1;
```

The screenshot shows the AWS Athena Query Editor interface. At the top, there are tabs for 'Athena' and 'Query Editor' (which is selected), along with links for 'Saved Queries', 'History', 'AWS Glue Data Catalog', and 'Workgroup : primary'.  
  
On the left, the 'Catalog' section shows 'Glue' selected under 'Database' and 'weblogs\_dev' selected under 'Tables'. It also includes a 'Filter tables and views...' input field and sections for 'Tables (2)' and 'Views (0)'.  
  
The main area contains a 'New query 1' tab and a 'New query 2' tab (highlighted with an orange border). Below these tabs is a code editor with the following SQL query:

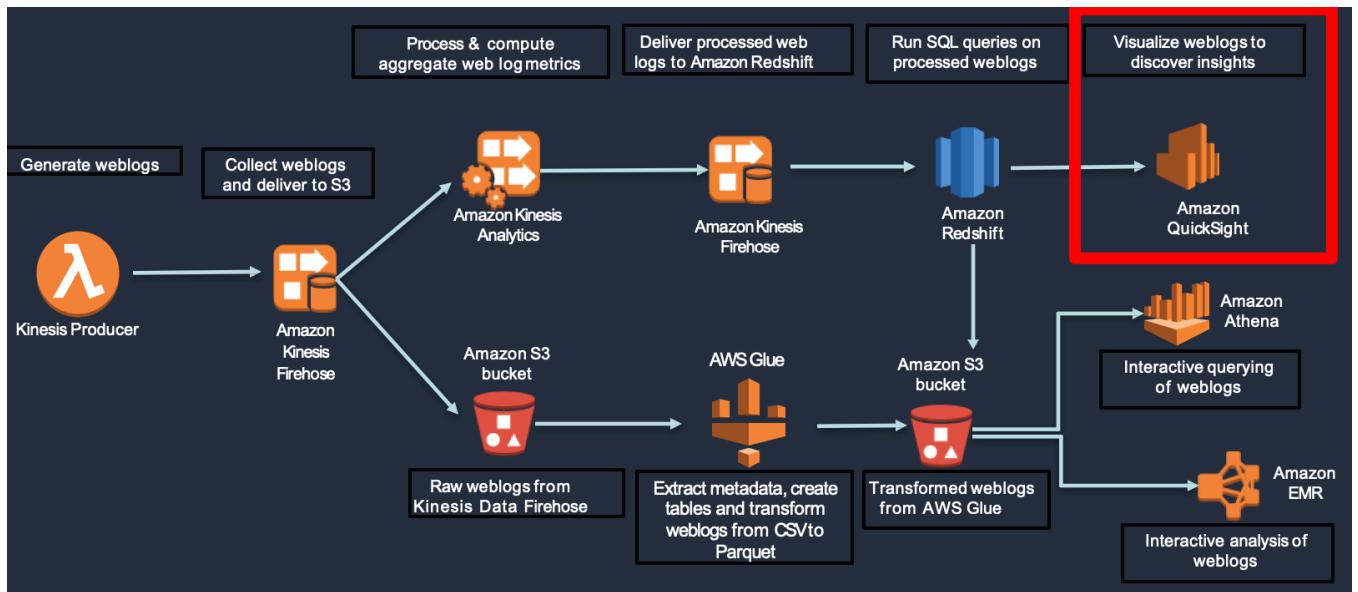
```
1 SELECT * FROM "weblogs_dev"."parquet" limit 10;
```

  
At the bottom of the interface, there are buttons for 'Run query', 'Save as', and 'Create' (with a dropdown menu), and a status message '(Run time: 3.33 seconds, Data scanned: 1.27)'. A note at the bottom says 'Use Ctrl + Enter to run query, Ctrl + Space to autocomplete'.

## Task 6 (optional): Data visualization with Amazon QuickSight (20 min)

We are going to:

- A. Register for an Amazon QuickSight account
- B. Connect to the Amazon Redshift cluster
- C. Create visualizations for analysis to answer questions like:
  - A. What are the most common http requests and how successful (response code of 200) are they?
  - B. Which are the most requested URIs?

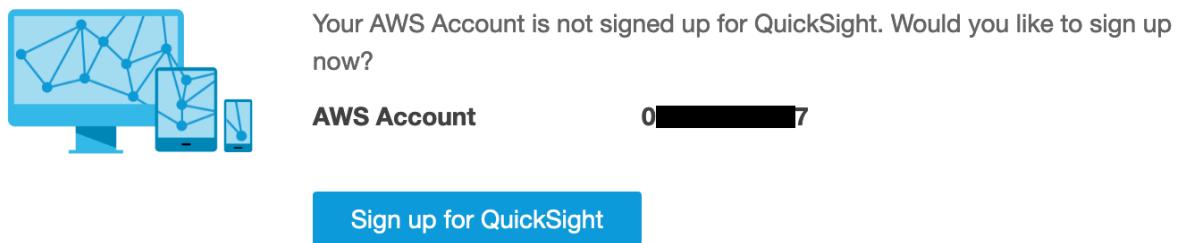


## 6.1 Amazon QuickSight registration

**Note:**

Follow step 6.1 - only if you have never used QuickSight in your AWS account.

- Go to AWS console, click on Amazon QuickSight from the analytics section



To access QuickSight with a different account, [log in](#) again.

- Click on "Sign up for QuickSight" in the next window
- Make sure the subscription type is standard and click "Continue" on the next screen

Create your QuickSight account

Edition Standard

---

QuickSight region  
Select a region. i  
US East (N. Virginia) ▼

---

QuickSight account name  
Enter a unique QuickSight account name i  
You will need this for you and others to sign in.

---

Notification email address  
Enter account notification email address i  
For QuickSight to send important notifications.

---

Enable autodiscovery of data and users in your Amazon Redshift, Amazon RDS, and AWS IAM services.

Amazon Athena  
Enables QuickSight access to Amazon Athena databases

Please ensure the right Amazon S3 buckets are also enabled for QuickSight.

---

Amazon S3 Choose S3 buckets  
Enables QuickSight to auto-discover your Amazon S3 buckets

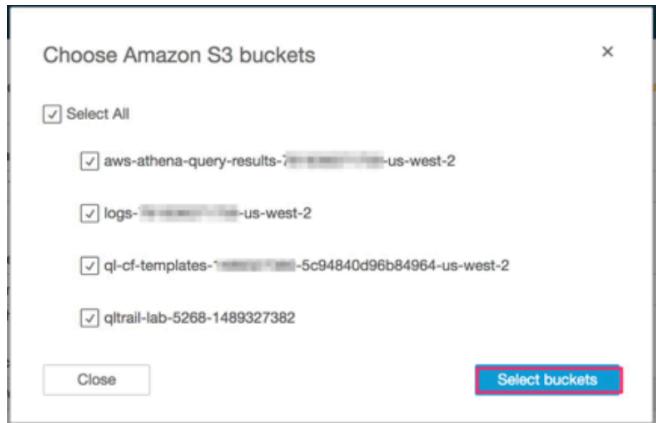
Amazon S3 Storage Analytics  
Enables QuickSight to visualize your S3 Storage Analytics data

AWS IoT Analytics  
Enables QuickSight to visualize your IoT Analytics data

---

Finish

- On the "Subscription type" page, enter the account name (see **note** below)
- Note: Amazon QuickSight account name is your AWS account number**
- Enter your **email address**
- Select < **AWS region**> where you created CloudFormation stack.
- Check the "**Amazon S3 (all buckets)**" box



- Click "Select buckets"
- Click "Finish".

## 6.2 Connect to Amazon Redshift (Data Store) for QuickSight

- Click on "Manage Data"

• select "**New Data set**" to create a new data set in Amazon QuickSight

- Choose "Redshift (Auto-discovered)" as the data source.

Note: Amazon QuickSight auto-discovers databases associated with your AWS account (Amazon Redshift database in this case).

- Use "dbadmin" as the username.
- You can get the Amazon Redshift database password by navigating to the AWS CloudFormation outputs section of Redshift Stack.

New Redshift data source ×

**Data source name**  
Redshift-weblogs

**Instance ID**  
bd-workshop-redshiftstack-155ndq4-redshiftcluster-igye97eqo0w2

**Connection type**  
Public network

**Database name**  
logs

**Username**  
dbadmin

**Password**  
.....

Validate connection    SSL is enabled    Create data source

- Click on "Validate connection". It will change status to Validated if entered details are correct.

✓ Validated    SSL is enabled    Create data source

- Click on “Create data source”.
- Choose your weblogs Amazon Redshift table
  - Select “public” schema
  - Select “weblogs” in tables

Choose your table X

Redshift-weblogs

**Schema: contain sets of tables.**

▼

**Tables: contain the data you can visualize.**

weblogs

---

Edit/Preview data Use custom SQL Select

- Click “Select”.
- Choose - Import to SPICE for quicker analytics

Finish data set creation X

Table: weblogs  
 Estimated table size: 100MB SPICE  
 Data source: Redshift-weblogs  
 Schema: public

---

Import to SPICE for quicker analytics ✓ 40.2GB available SPICE

---

Directly query your data

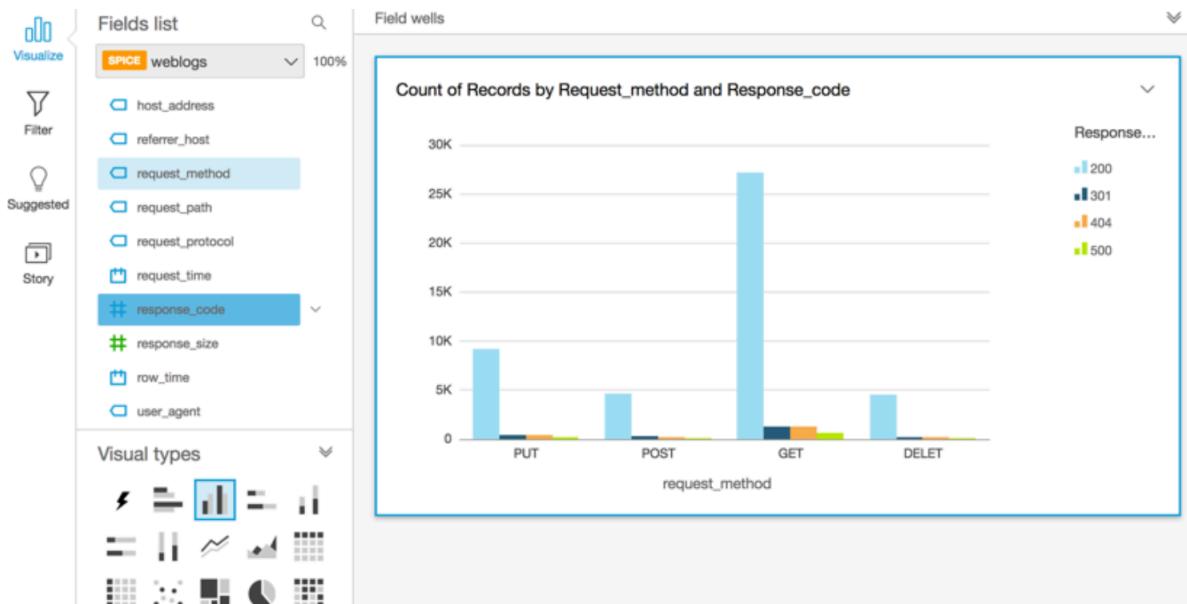
---

Edit/Preview data Visualize

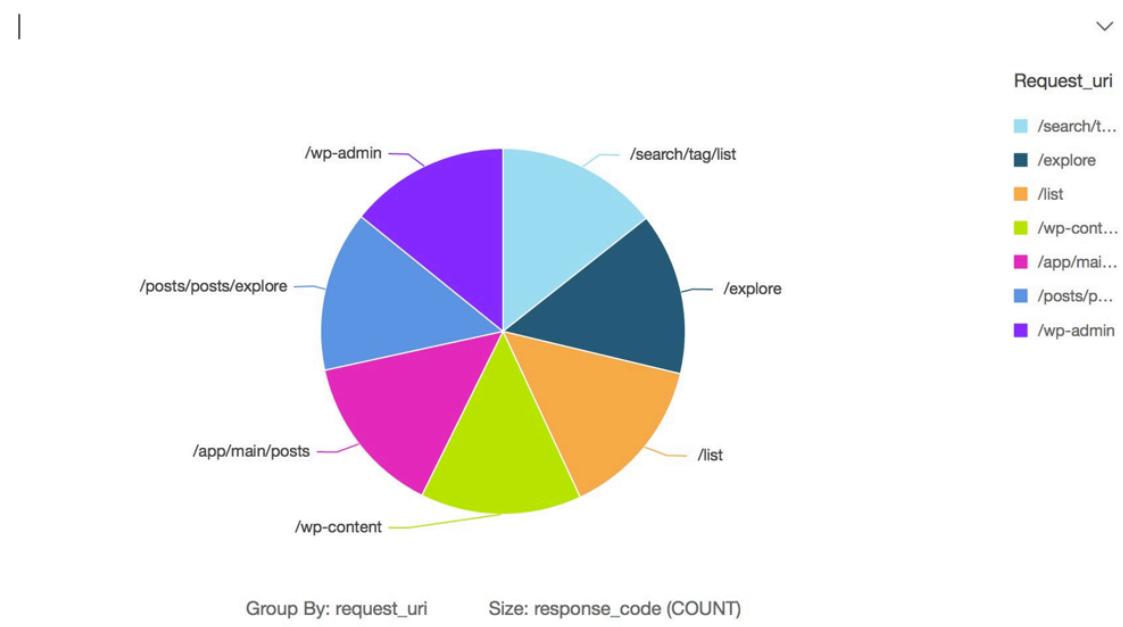
- Click “Visualize”.

## 6.3 Creating your first analysis

- What are the most requested http request types and their corresponding response codes for this site?
- Simply select request, response, and let AUTOGRAPH create the optimal visualization



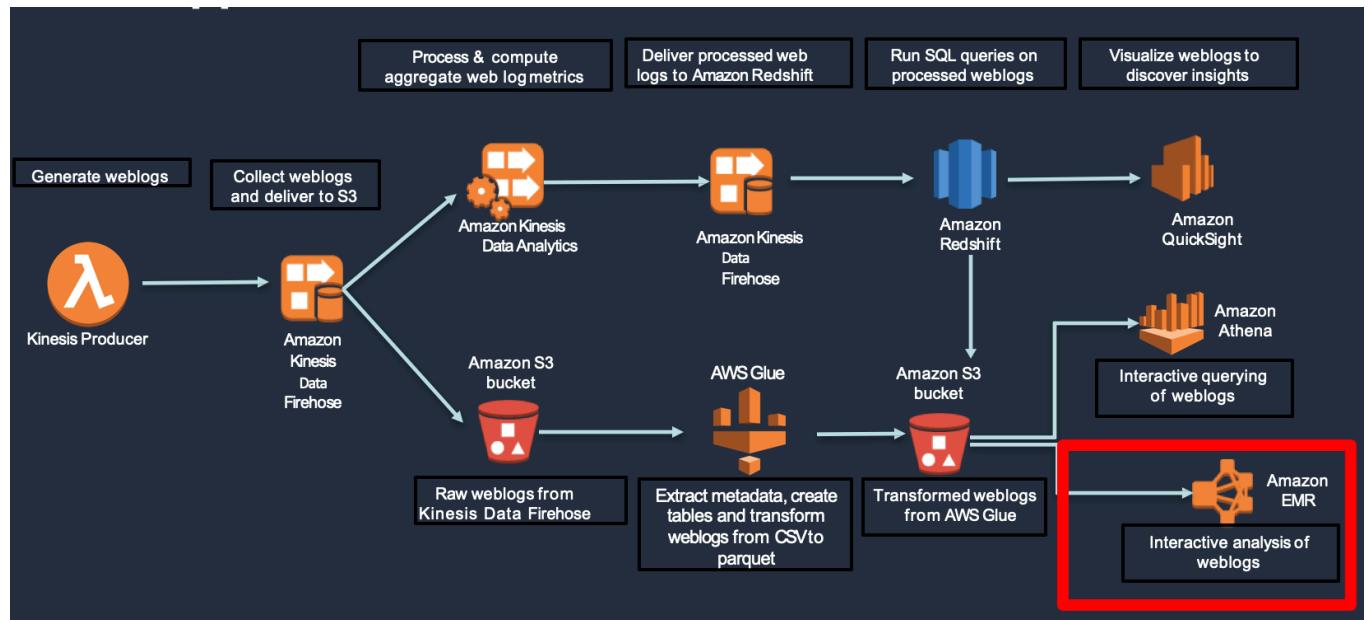
- Add a visual to demonstrate which URI are the most requested



## Task 7 (Optional): Interactive analysis using Amazon EMR (20 min)

We are going to:

- Use a Zeppelin notebook to interact with Amazon EMR cluster
- Process the data in Amazon S3 using Apache Spark
- Query the data processed in the earlier stage and create simple charts



### 7.1 Open the Zeppelin interface

- Copy the Zeppelin end point in the **AWS CloudFormation** output section of nested stack <stack-name>-EmrStack-<xxxxx>

The screenshot shows the AWS CloudFormation console with the 'Outputs' tab selected for the stack 'bd-workshop-EmrStack-13HOX5RMQ9OZE'. The 'Outputs' table lists one output: 'EmrMasterNodeDns' with the value 'ec2-34-205-39-95.compute-1.amazonaws.com'. The left sidebar shows a list of other stacks in the hierarchy, including 'bd-workshop-KinesisStack-1C2IWUSWPTVS' and 'bd-workshop-GlueStack-14UJHX0DS5AXB', both of which are marked as 'CREATE\_COMPLETE'.

- Open the Zeppelin link in a new browser window

The screenshot shows the Zeppelin homepage. At the top, there's a blue header bar with the Zeppelin logo and a "Notebook" dropdown. Below the header, a large banner says "Welcome to Zeppelin!". It includes a brief introduction: "Zeppelin is web-based notebook that enables interactive data analytics. You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!" On the left, there's a sidebar titled "Notebook" with options like "Import note" (which is highlighted with a red box), "Create new note", and a search bar. On the right, there are sections for "Help" (with links to documentation), "Community" (with links to a mailing list, issues tracking, and GitHub), and a large decorative graphic of a stylized aircraft.

- Open the First Big Data Application json file from the BigDataWorkshop.zip
  - Import the notebook using the "Import Note" link on Zeppelin interface
- Note: You may need to disconnect from VPN or the page will not load**
- Execute Step 1 - Run the first paragraph
    - Enter <stackname>-logs-<account>-us-west-2/processed/parquet in the Bucket field as input
    - This is where the processed parquet files were stored earlier in your S3 bucket

The screenshot shows a Zeppelin notebook titled "First Big Data Application". The notebook interface has a toolbar at the top with various icons. The main area contains a step titled "Step 1 : Get the Bucket name". It shows a Scala code snippet: 

```
val bucketName = z.input("Bucket")
```

. Below the code, a "Bucket" field contains the value "qls-108881-f75177905b7b5b0e-logs-". The output section shows the result: 

```
bucketName: Object = qls-108881-f75177905b7b5b0e-logs- -us-west-2/weblogs/processed/parquet
```

. At the bottom, it says "Took 1 sec. Last updated by anonymous at November 23 2017, 9:51:06 PM." A red arrow points to the "Run Paragraph" button in the toolbar.

## 7.2 Run the notebook

- Execute Step 2
  - Create a data frame with the parquet files from the AWS Glue ETL job
- Execute Step 3
  - Sample a few rows

Zeppelin Notebook -

First Big Data Application

Step 2 : Read data from S3 and create an in-memory Dataframe

```

import org.apache.spark.sql.SQLContext
val sqlContext = new SQLContext(sc)
val weblogsPath = "s3://<your-s3-bucket>/weblogs/processed.parquet"
val weblogsDF = sqlContext.read.parquet(weblogsPath)
weblogsDF.show()

```

FINISHED

Step 3 : Sample a few rows

```

weblogsDF.show()
weblogsDF.printSchema()

```

FINISHED

clientip	verb	request[httpversion]	agent	dt responses bytes
238.108.161.6	GET	/explore	1.1 Mozilla/5.0 (comp... 2018-07-02 00:00:...	200  9313
143.254.201.146	GET	/app/main/posts	1.1 Mozilla/5.0 (Wind... 2018-07-02 00:00:...	200  4995
171.157.56.48	GET	/search/tag/list	1.1 Mozilla/5.0 (comp... 2018-07-02 00:00:...	200  7043
213.199.47.142	PUT	/explore	1.1 Mozilla/5.0 (Mac... 2018-07-02 00:00:...	301  4095
142.224.43.56	GET	/wp-admin	1.1 Mozilla/5.0 (comp... 2018-07-02 00:00:...	200  9225
2.148.133.123	PUT	/search/tag/list	1.1 Mozilla/5.0 (comp... 2018-07-02 00:00:...	200  287
126.32.233.51	PUT	/wp-admin	1.1 Mozilla/5.0 (comp... 2018-07-02 00:00:...	200  3211
81.9.64.26	DELETE	/wp-admin	1.1 Mozilla/5.0 (Wind... 2018-07-02 00:00:...	200  6653
229.94.119.172	GET	/posts/posts/explore	1.1 Mozilla/5.0 (comp... 2018-07-02 00:00:...	200  8629

Took 17 sec. Last updated by anonymous at July 05 2018, 2:21:23 PM.

- Execute Step 4 to process the data
  - Notice how the “AGENT” field consists of the “BROWSER” at the beginning of the column value. Let’s extract the browser from the agent field.
- Create a UDF to extract the browser and add to the data frame
- Print the new data frame

Zeppelin Notebook -

First Big Data Application

Step 4: Process the 'AGENT' column and extract the 'BROWSER' portion

```

val extractColumns = udf { (s: String) => s.substring(0, s.indexOf("/")).trim }

val newWeblogsDF = weblogsDF.withColumn("browser", extractColumns($"AGENT"))

newWeblogsDF.printSchema()

newWeblogsDF.show()

```

extractColumns: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function1>,StringType,Some(List(StringType)))
newWeblogsDF: org.apache.spark.sql.DataFrame = [clientip: string, verb: string ... 7 more fields]
root
 |-- clientip: string (nullable = true)
 |-- verb: string (nullable = true)
 |-- request: string (nullable = true)
 |-- httpversion: string (nullable = true)
 |-- agent: string (nullable = true)
 |-- dt: timestamp (nullable = true)
 |-- response: integer (nullable = true)
 |-- bytes: integer (nullable = true)
 |-- browser: string (nullable = true)

clientip	verb	request[httpversion]	agent	dt response bytes browser
238.108.161.6	GET	/explore	1.1 Mozilla/5.0 (comp... 2018-07-02 00:00:...	200  9313 Mozilla
143.254.201.146	GET	/app/main/posts	1.1 Mozilla/5.0 (Wind... 2018-07-02 00:00:...	200  4995 Mozilla
171.157.56.48	GET	/search/tag/list	1.1 Mozilla/5.0 (comp... 2018-07-02 00:00:...	200  7043 Mozilla
213.199.47.142	PUT	/explore	1.1 Mozilla/5.0 (Mac... 2018-07-02 00:00:...	301  4095 Mozilla
142.224.43.56	GET	/wp-admin	1.1 Mozilla/5.0 (comp... 2018-07-02 00:00:...	200  9225 Mozilla
2.148.133.123	PUT	/search/tag/list	1.1 Mozilla/5.0 (comp... 2018-07-02 00:00:...	200  287 Mozilla
126.32.233.51	PUT	/wp-admin	1.1 Mozilla/5.0 (comp... 2018-07-02 00:00:...	200  3211 Mozilla
81.9.64.26	DELETE	/wp-admin	1.1 Mozilla/5.0 (Wind... 2018-07-02 00:00:...	200  6653 Mozilla
229.94.119.172	GET	/posts/posts/explore	1.1 Mozilla/5.0 (comp... 2018-07-02 00:00:...	200  8629 Mozilla
38.67.72.36	GET	/list	1.1 Mozilla/5.0 (Wind... 2018-07-02 00:00:...	200  3535 Mozilla
112.193.177.232	GET	/explore	1.1 Mozilla/5.0 (Wind... 2018-07-02 00:00:...	200  8557 Mozilla

Took 5 sec. Last updated by anonymous at July 05 2018, 2:28:45 PM.

- Execute Step 6
  - Register the data frame as a temporary table

- Now you can run SQL queries on the temporary tables



- Execute the next 3 steps and observe the charts created
- What did you learn about the data set?

#### Note:

- You just learned on how to process and query data using Amazon EMR with Apache Spark
- Amazon EMR has many other frameworks available for you to use Hive, Presto, Flink, Pig, MapReduce, Hue, Oozie, HBase

## Delete all resources

Make sure you delete the Replication instance. Once the replication instance is deleted you should be able to delete the CloudFormation stack, which will in turn delete all resources.

*Note: If you do not delete resources they will keep incurring charges.*

1. Open the **CloudFormation** console at:  
[https://console.aws.amazon.com/cloudformation/  
home](https://console.aws.amazon.com/cloudformation/home)
2. Select the checkbox for the stack you created for this lab: **bd-workshop**
3. Select **Actions** and **Delete Stack**. Select **Yes, Delete**. CloudFormation will delete all resources created.