Workshop

Building Your First Big Data Application on AWS

Table of Contents

Prerequisites

In order to complete this workshop, you'll need

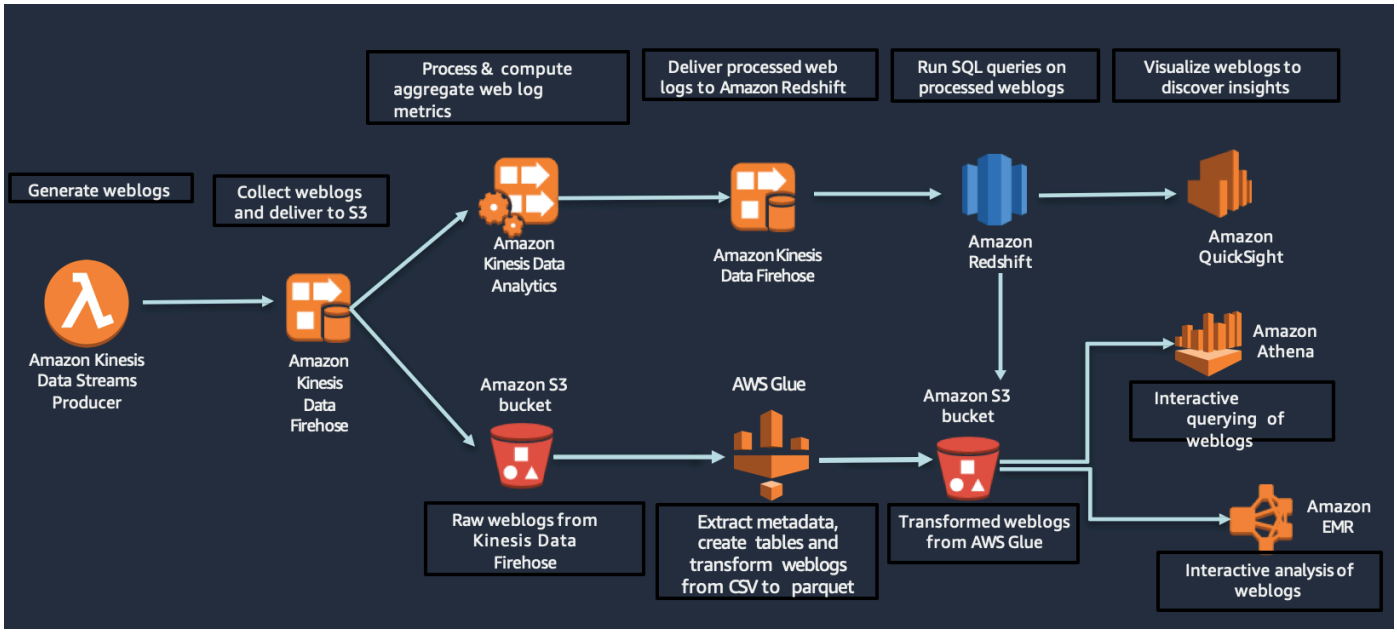
- An active AWS Account,
- An AWS IAM user in that account with at least full permissions to the following AWS services:
    - AWS IAM
    - Amazon S3
    - Amazon Kinesis
    - AWS Glue
    - Amazon Redshift
    - Amazon SageMaker

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

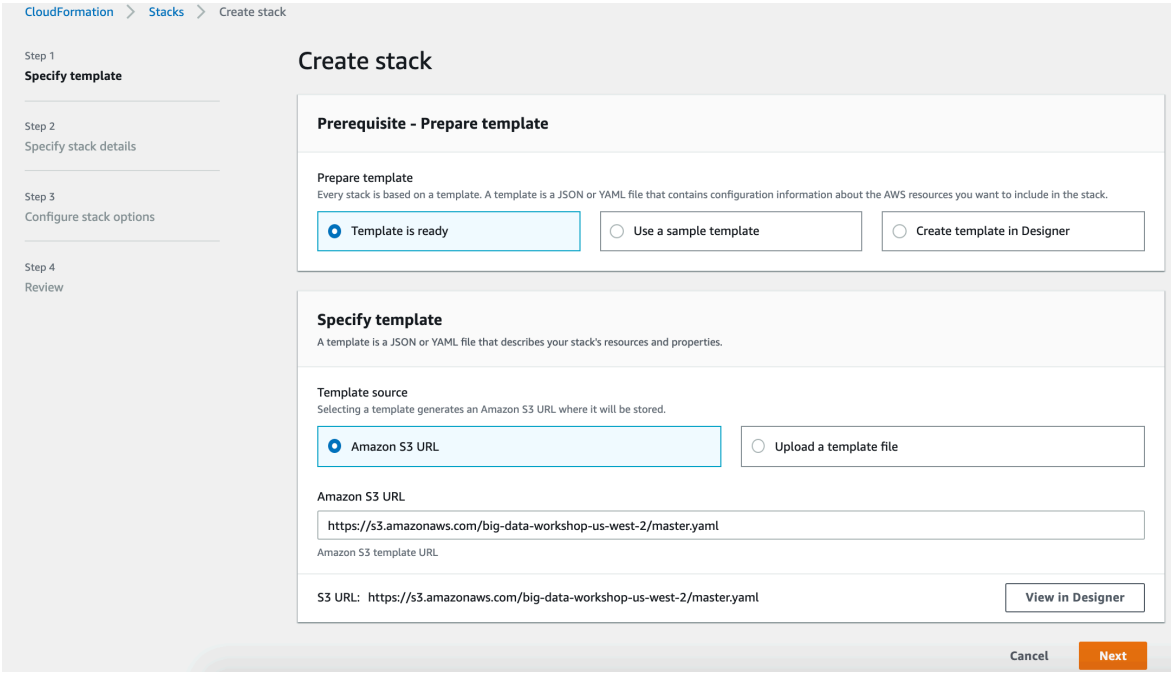
- Download BigDataWorkshop.zip, if not already done,
  Location: https://github.com/wrbaldwin/da-week/tree/master/Labs/First-Big-Data-App
  Or
  Use direct link: Workshop Scripts
- Unzip file BigDataWorkshop.zip.
- You will need these scripts during workshop.

Launch CloudFormation template to setup environment

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

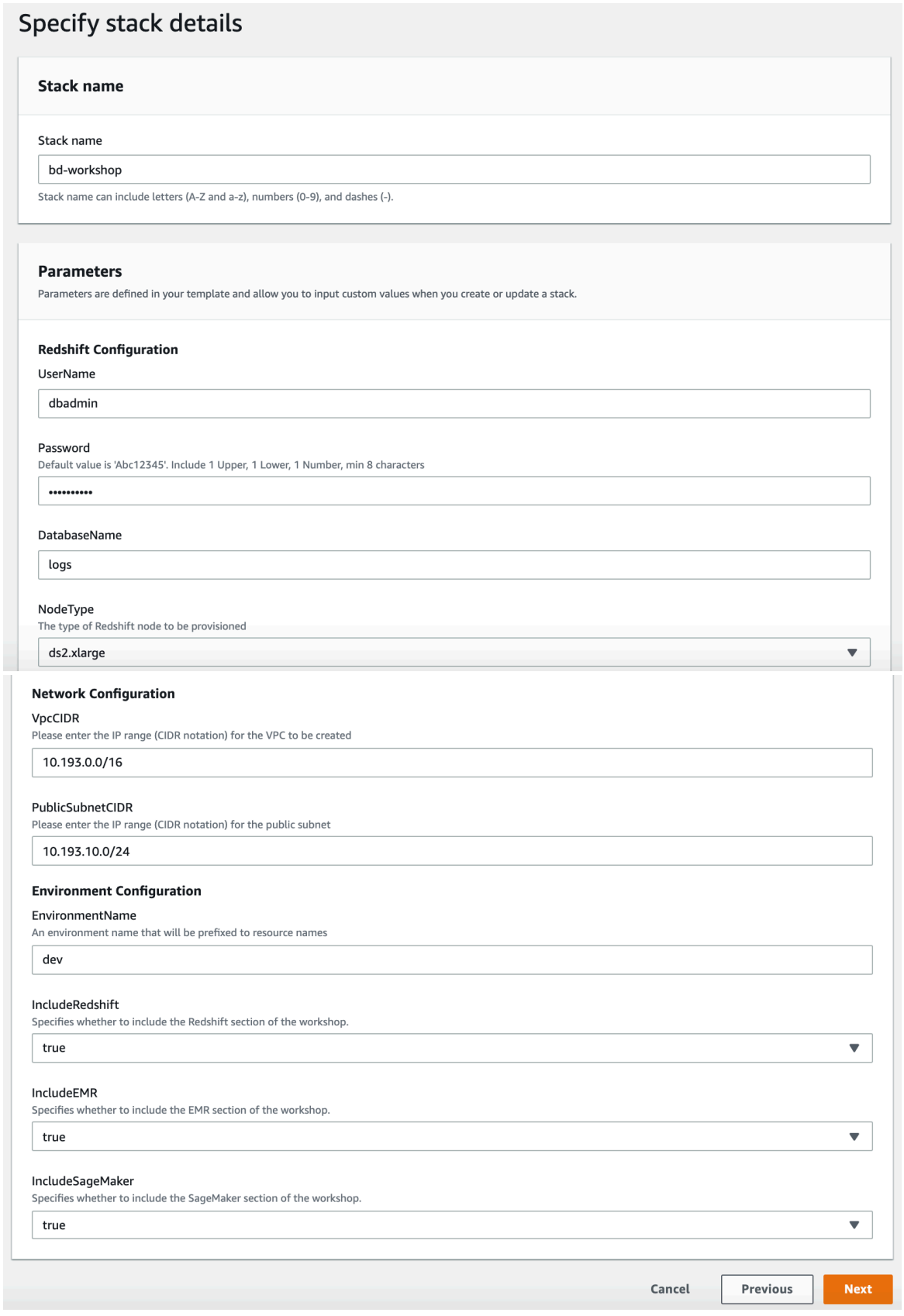
1. You can launch this CloudFormation stack in your account in one of the supported Regions below. Please click on “Launch Stack” based on your choice.

AWS Region	Short name	New VPC	Existing VPC
US East (N. Virginia)	us-east-1	Launch Stack	Launch Stack
US West (Oregon)	us-west-2	Launch Stack	Launch Stack
EU (Ireland)	eu-west-1	Launch Stack	Launch Stack
EU (Frankfurt)	eu-central-1	Launch Stack	Launch Stack

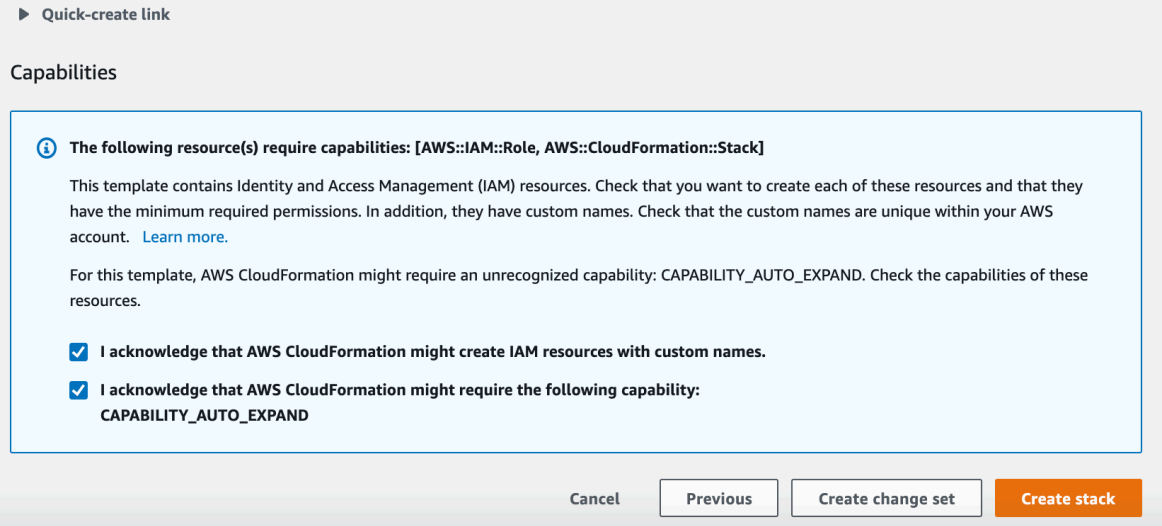


2. Verify the Amazon Simple Storage Service (Amazon S3) template URL, and click “Next”
3. Enter Stack Name: **bd-workshop** (if multiple users are using same account and same region – include suffice to keep name unique)
4. Enter other parameters as shown below:
    a. All defaults can be left as-is 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

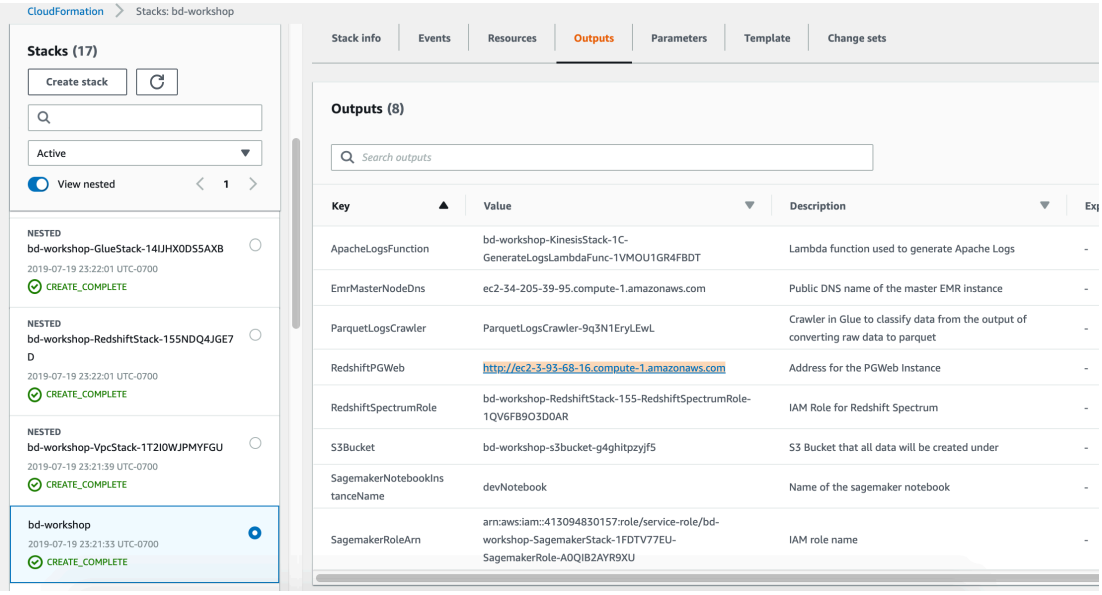
Abc12345 as the password. Make a note of this for later.



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’**



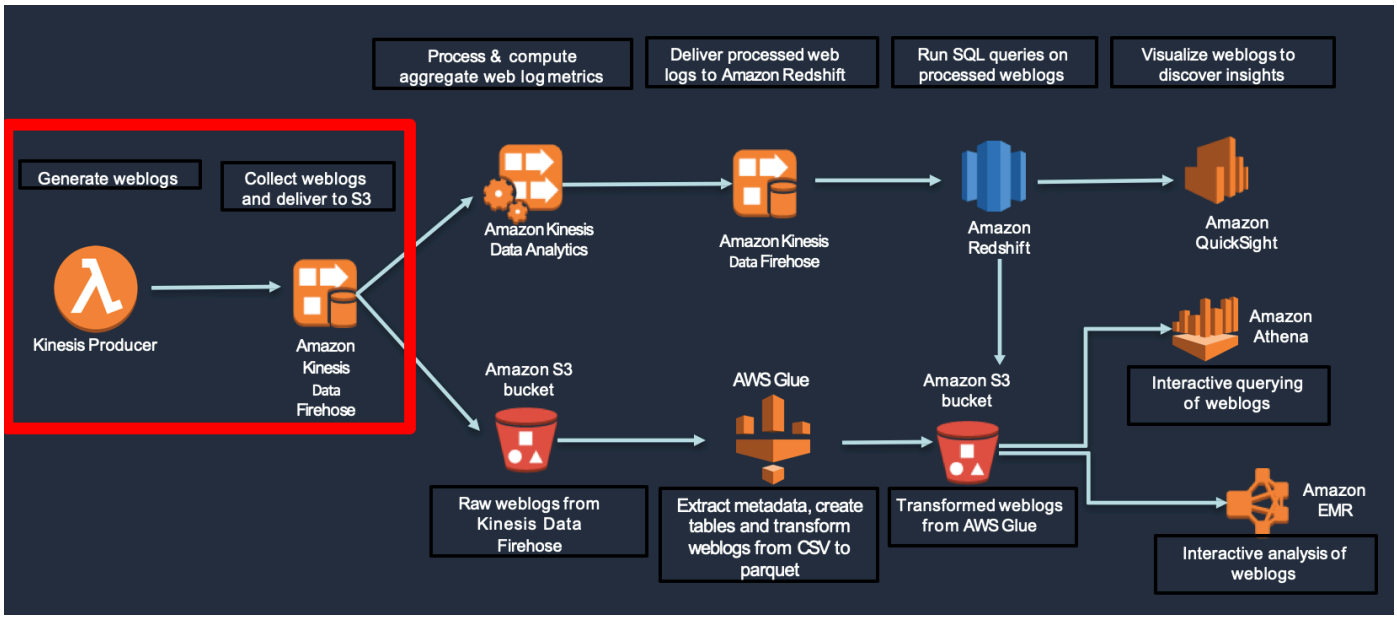
10. Click **Create Stack**.
11. At this point, you will be directed back to the CloudFormation console and will see a status of **CREATE_IN_PROGRESS**. Do not continue until the status changes to **CREATE_COMPLETE**.
    Note: this step may take 15-20 min to complete.



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

In this step, we are going to:

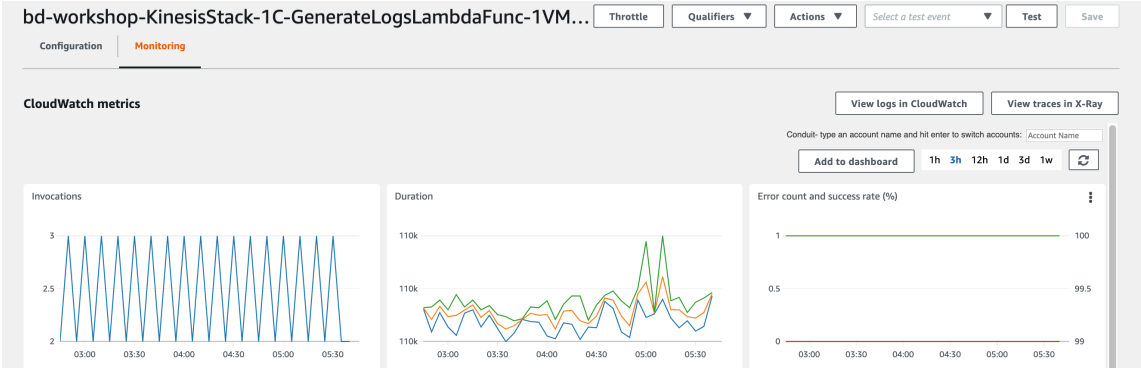
- Write to a Data Firehose delivery stream—Simulate writing transformed Apache weblogs to a Data Firehose delivery stream that is configured to deliver data into an S3 bucket.
- There are **many** different tools and libraries that can be used to write data to a Data Firehose delivery stream. One popular 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-xxxxxxx

1.1.2 Go to the Monitoring tab

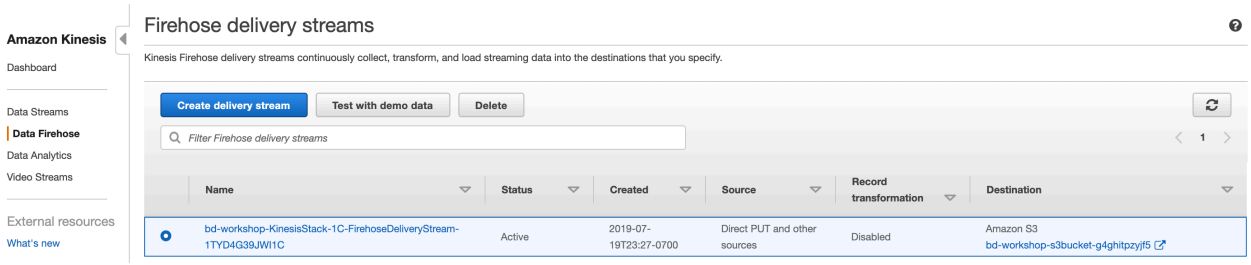


1.1.3 Click the “View logs in CloudWatch” button

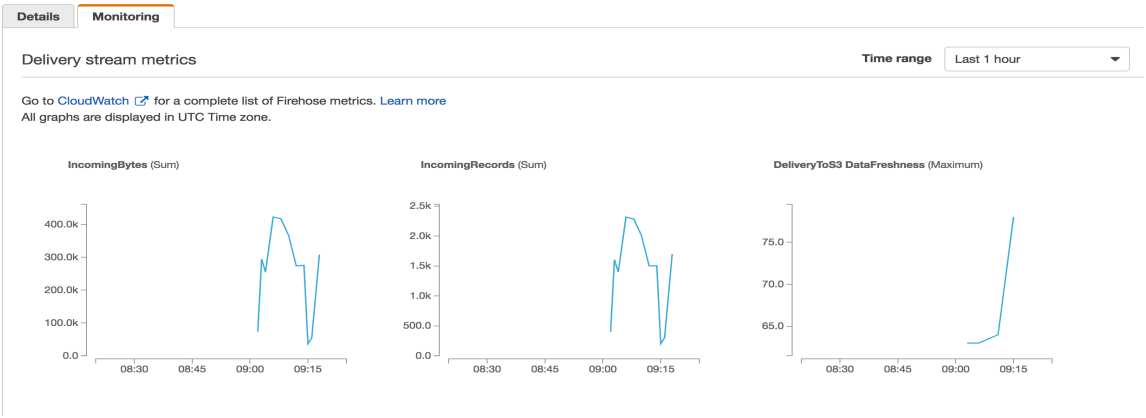
1.1.4 View the top log stream and you should see Apache log entries like below:

1.2 Monitoring Kinesis Data Firehose delivery to Amazon S3

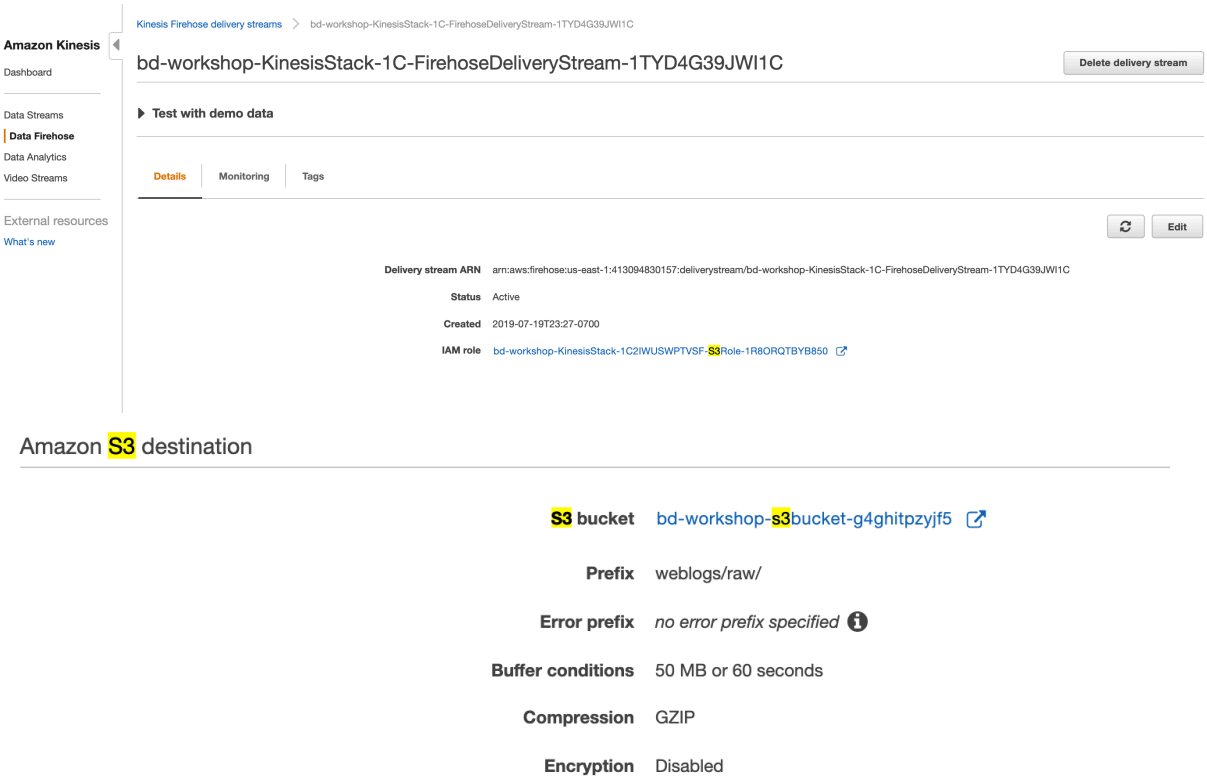
1.2.1 Go to the [Kinesis console](), click on “**Data Firehose**” and find data stream named like: <StackName>-KinesisStack-xxx-FirehoseDeliveryStream-xxxxxxx



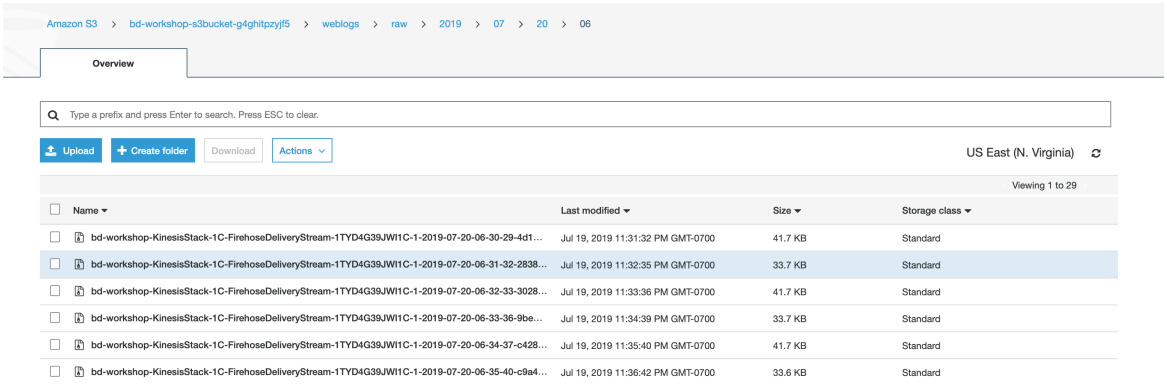
1.2.2 Select <StackName>-KinesisStack-xxx-FirehoseDeliveryStream-xxxxxxx
Go to the **Monitoring** tab.
There should be metrics for delivery to Amazon S3. This might take some time to show up.



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



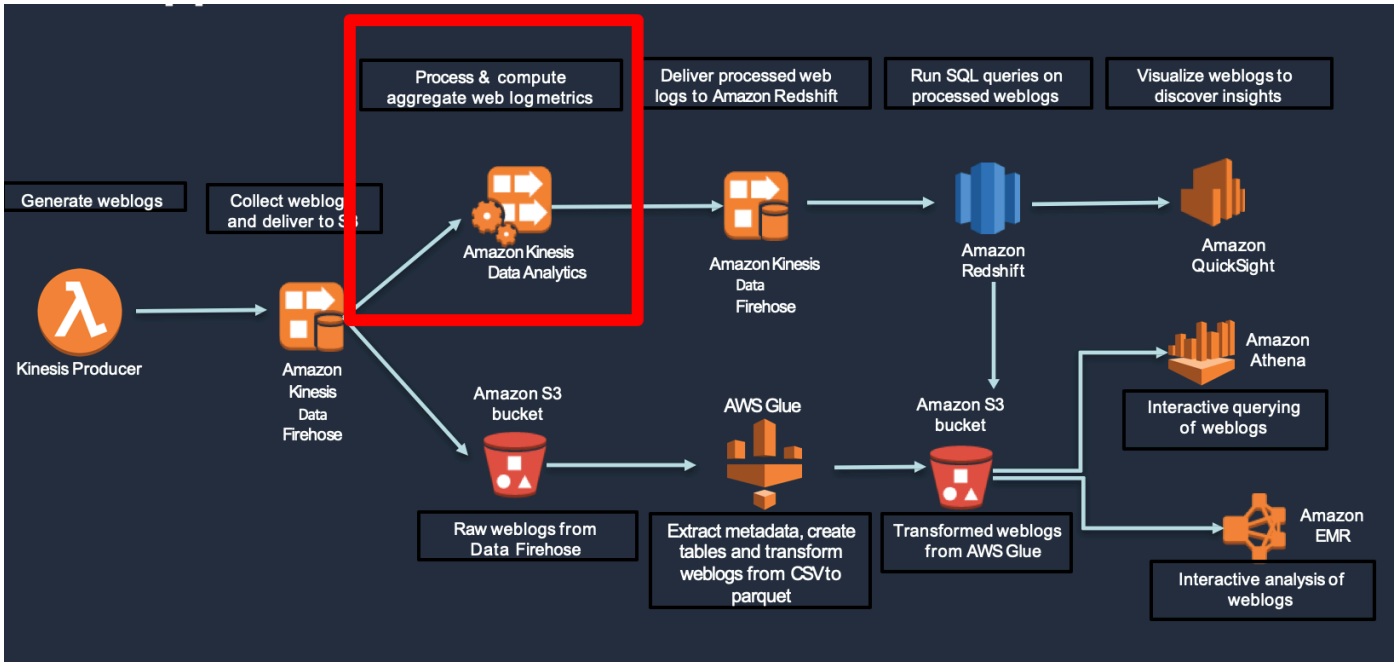
1.2.4 Click on S3 bucket and you can see that Apache logs are being streamed to this bucket:



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

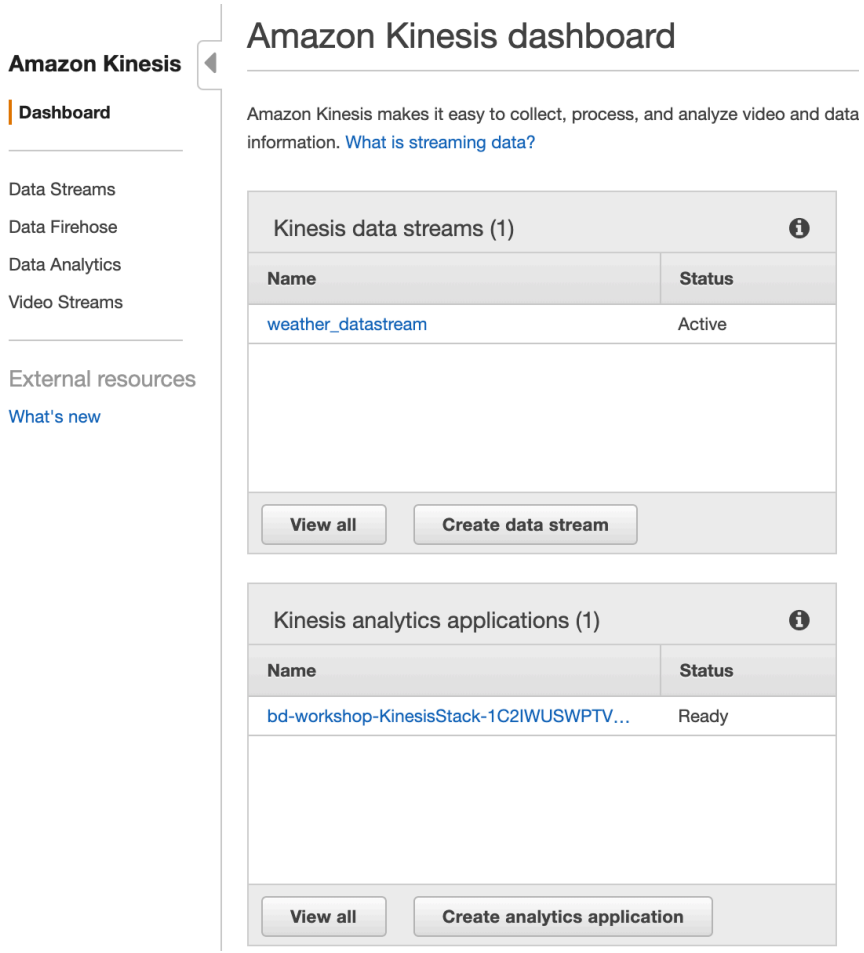
We are going to:

- 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

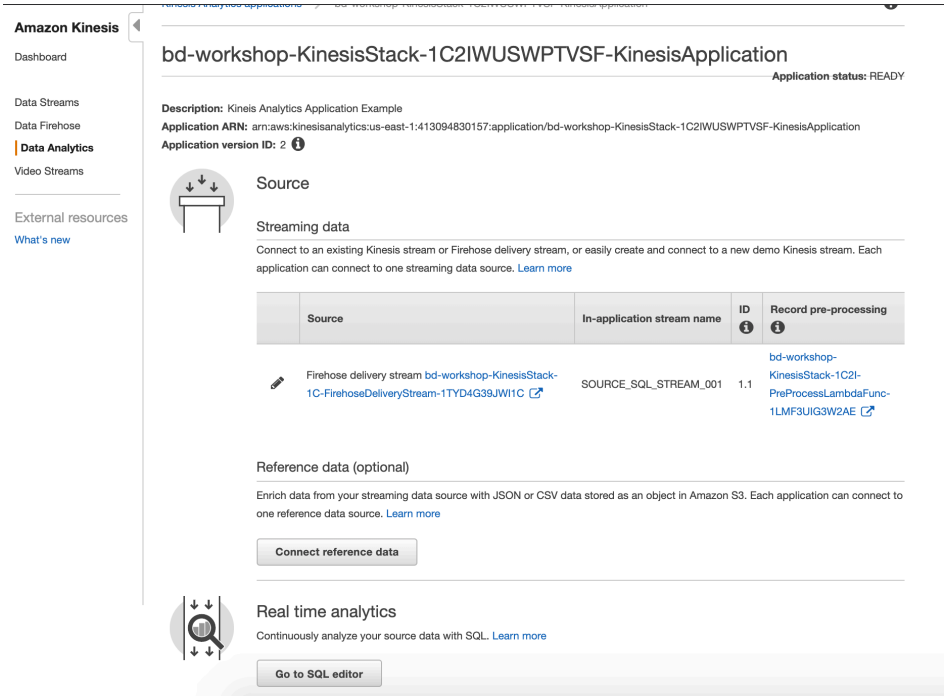


2.1 Start Amazon Kinesis Data Analytics app

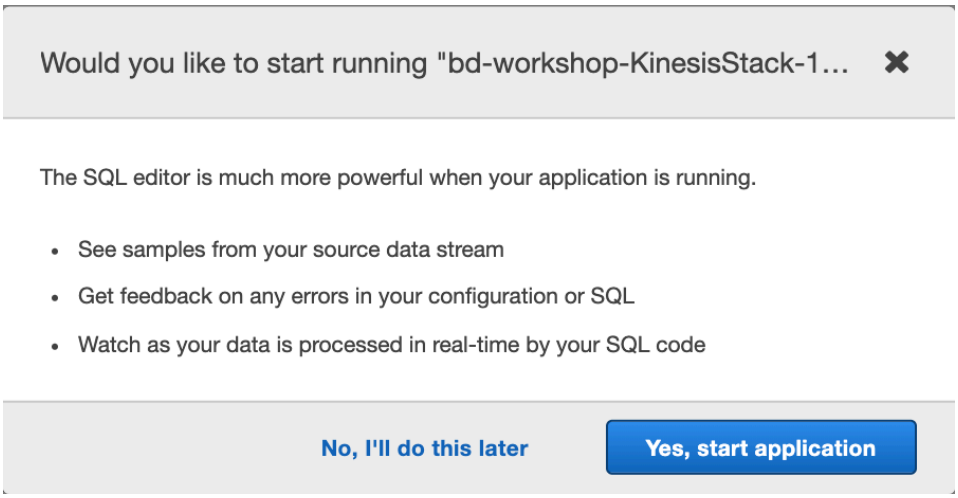
- Navigate to the **Kinesis** console,
- Click on the Dashboard and then **Kinesis Analytics application**



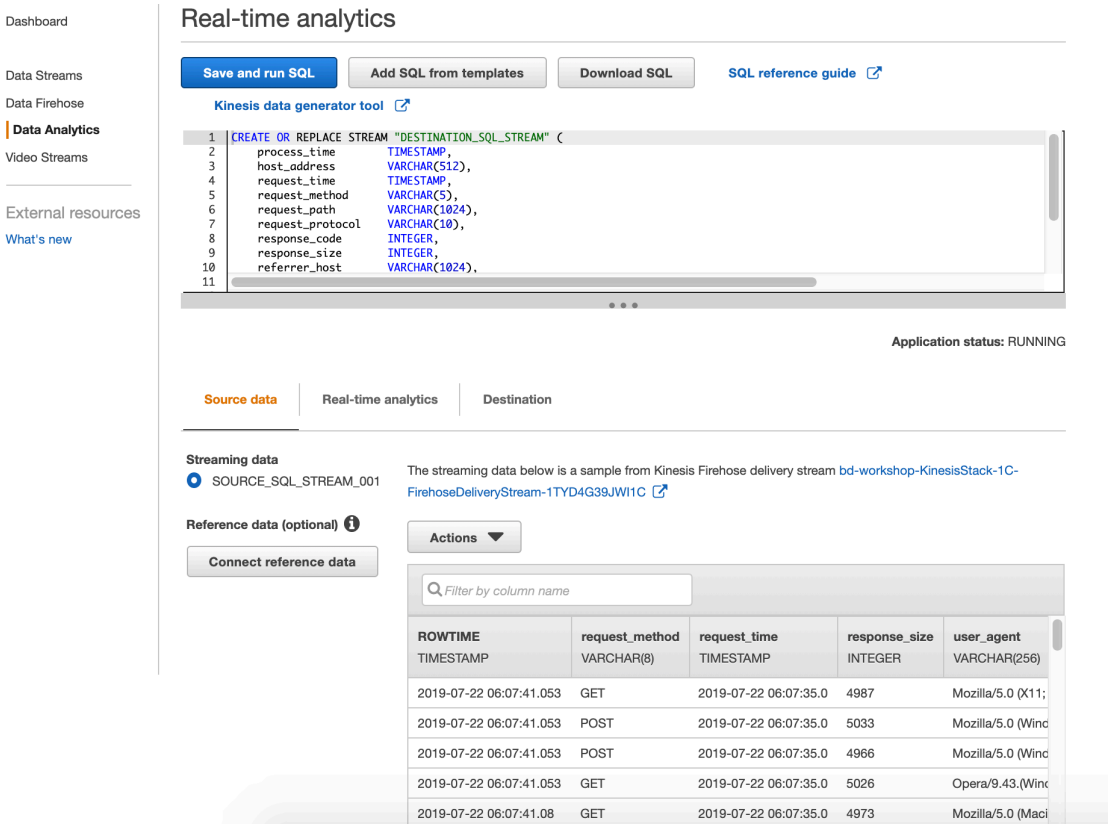
- Click on <stack-name>-KinesisStack-xxxxx-KinesisApplication



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

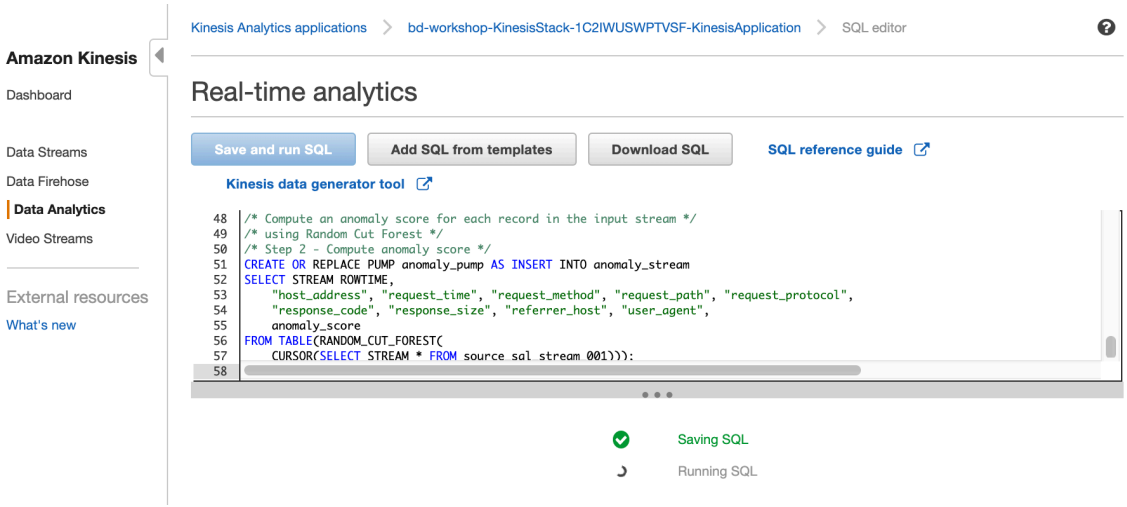


- View sample records in Kinesis app
- Review sample records delivered to the source stream (SOURCE_SQL_STREAM_001)

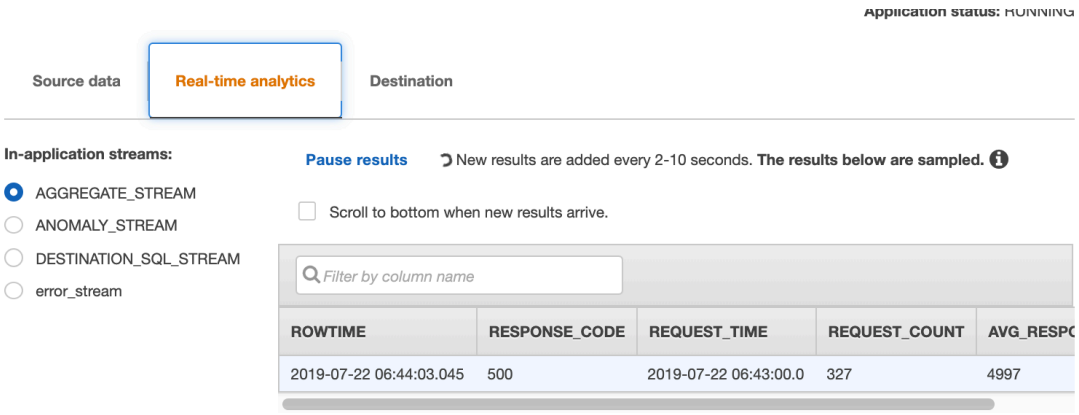


2.1 Calculate an aggregate metric

- 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**"



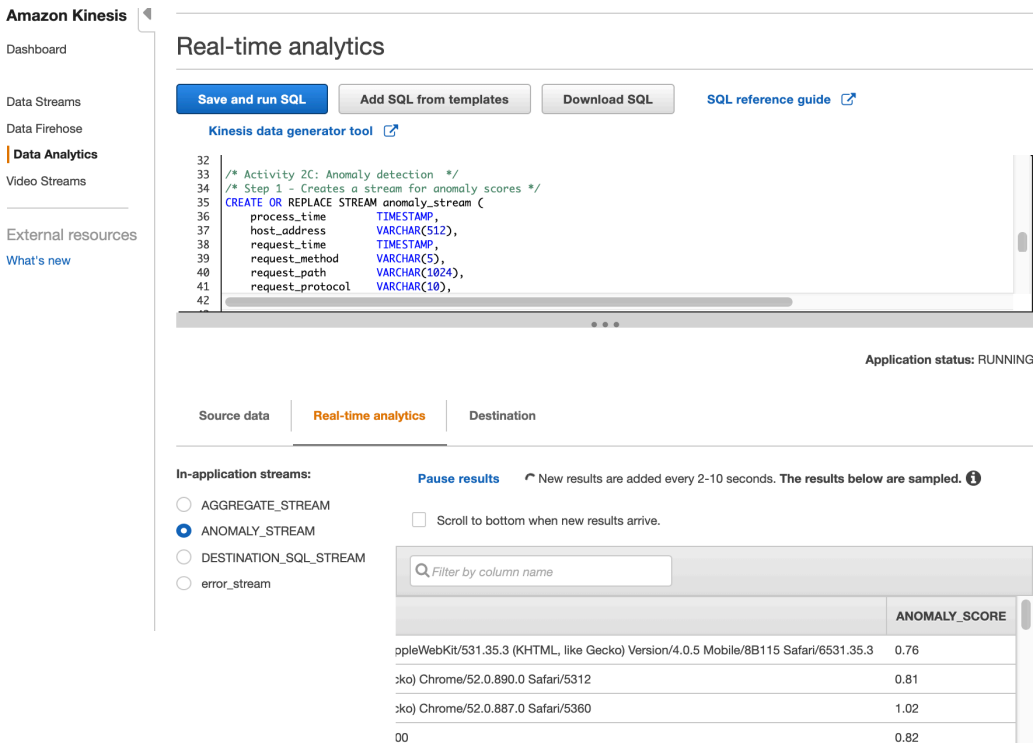
- This will run for few seconds and will produce results.



2.2 Anomaly detection

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

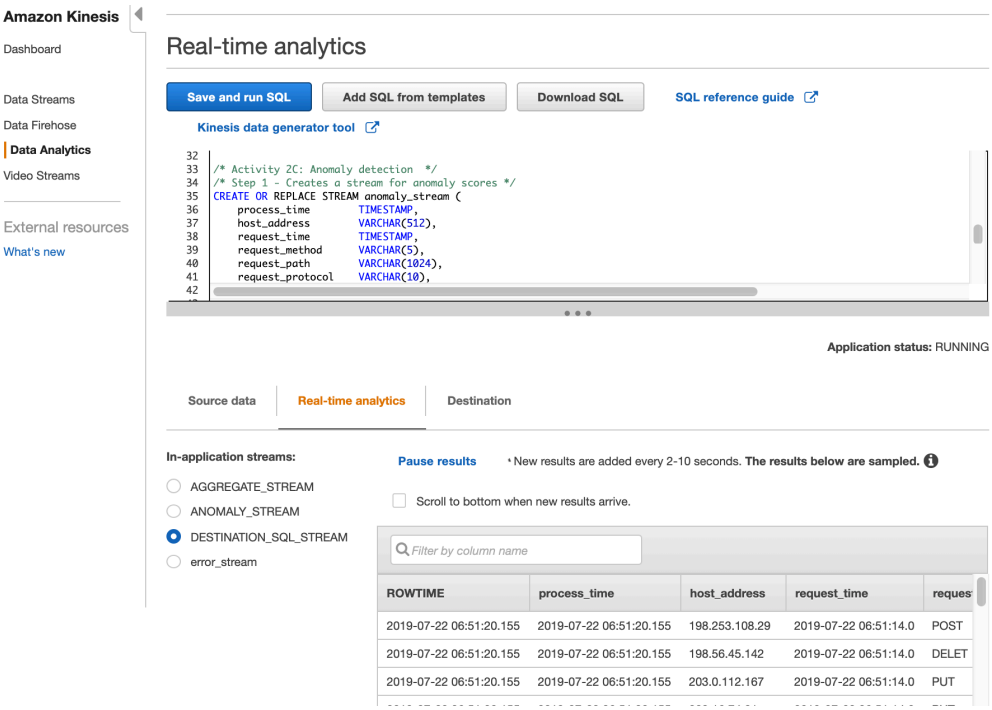
- It creates an ANOMALY_STREAM with the attribute anomaly_score
- It calculates the anomaly score for each record in the stream by using the built-in random cut forest function



2.3 Kinesis Data Analytics in-application streams

In-application streams:

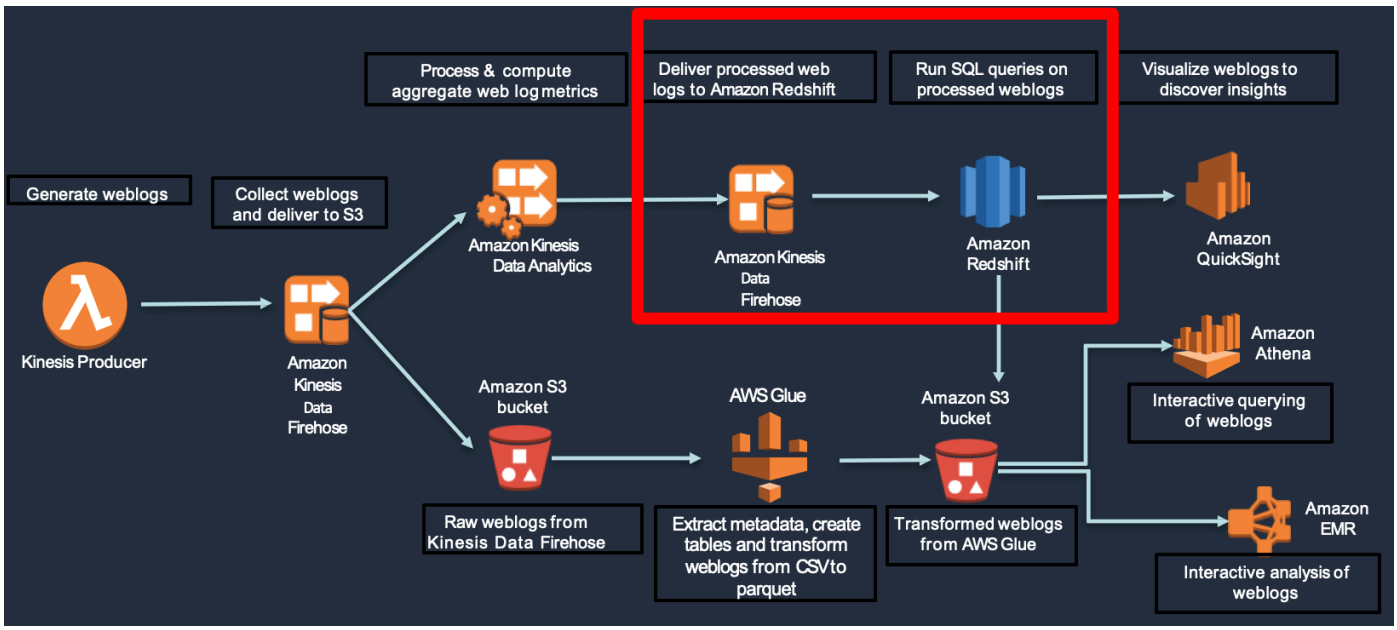
- Aggregate stream
- Anomaly stream
- Destination SQL stream
- Error stream



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

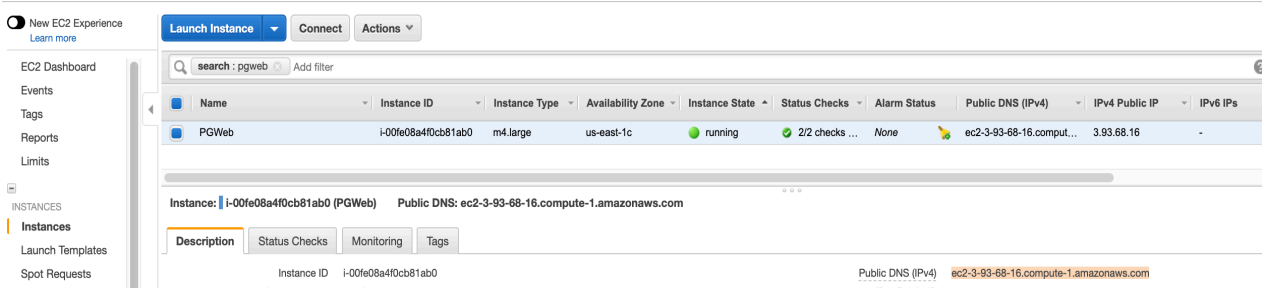
You can connect to Amazon Redshift cluster using one of below option

- 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



- 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
```

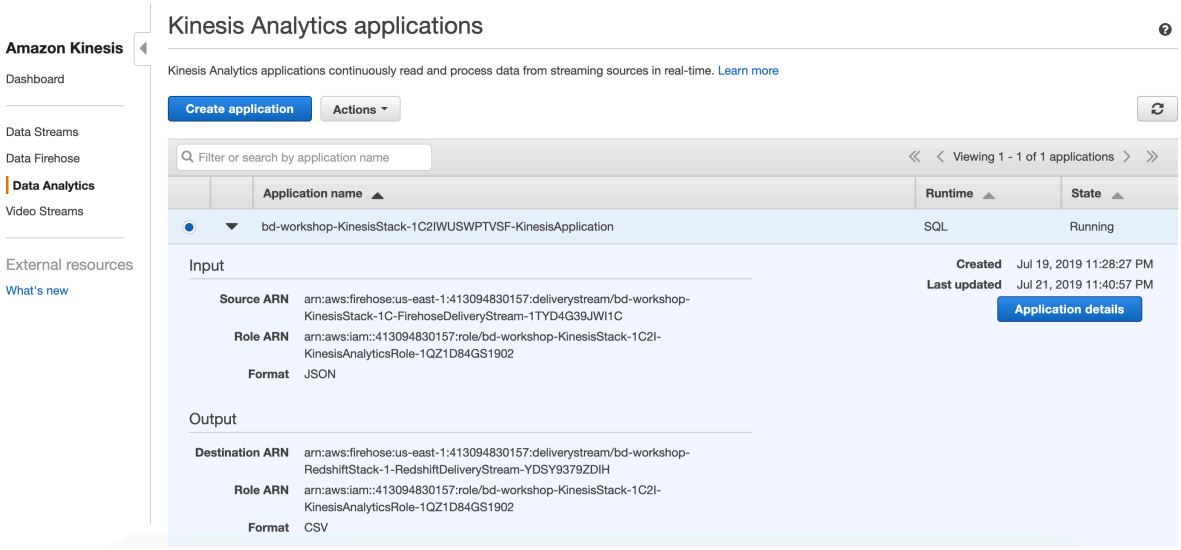
```
SORTKEY (request_time);
```

3.3 Deliver data to Amazon Redshift using Data Firehose

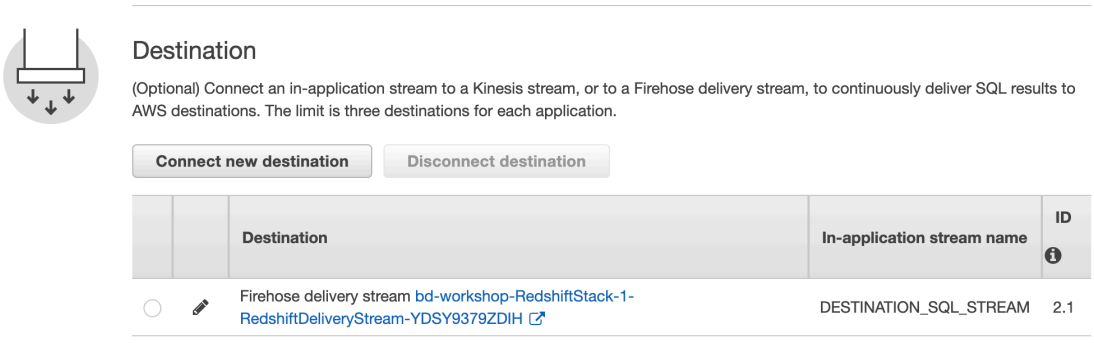
Update Kinesis Data Analytics application to send data to Kinesis Data Firehose delivery stream. Data Firehose delivers the streaming data to Amazon Redshift.

3.3.1 Go to the **Kinesis Data Analytics console**.

3.3.2 Select the KinesisApplication



3.3.3 Click on Application Details and scroll down until you see “Destination”.



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**-xxxxxxxx”

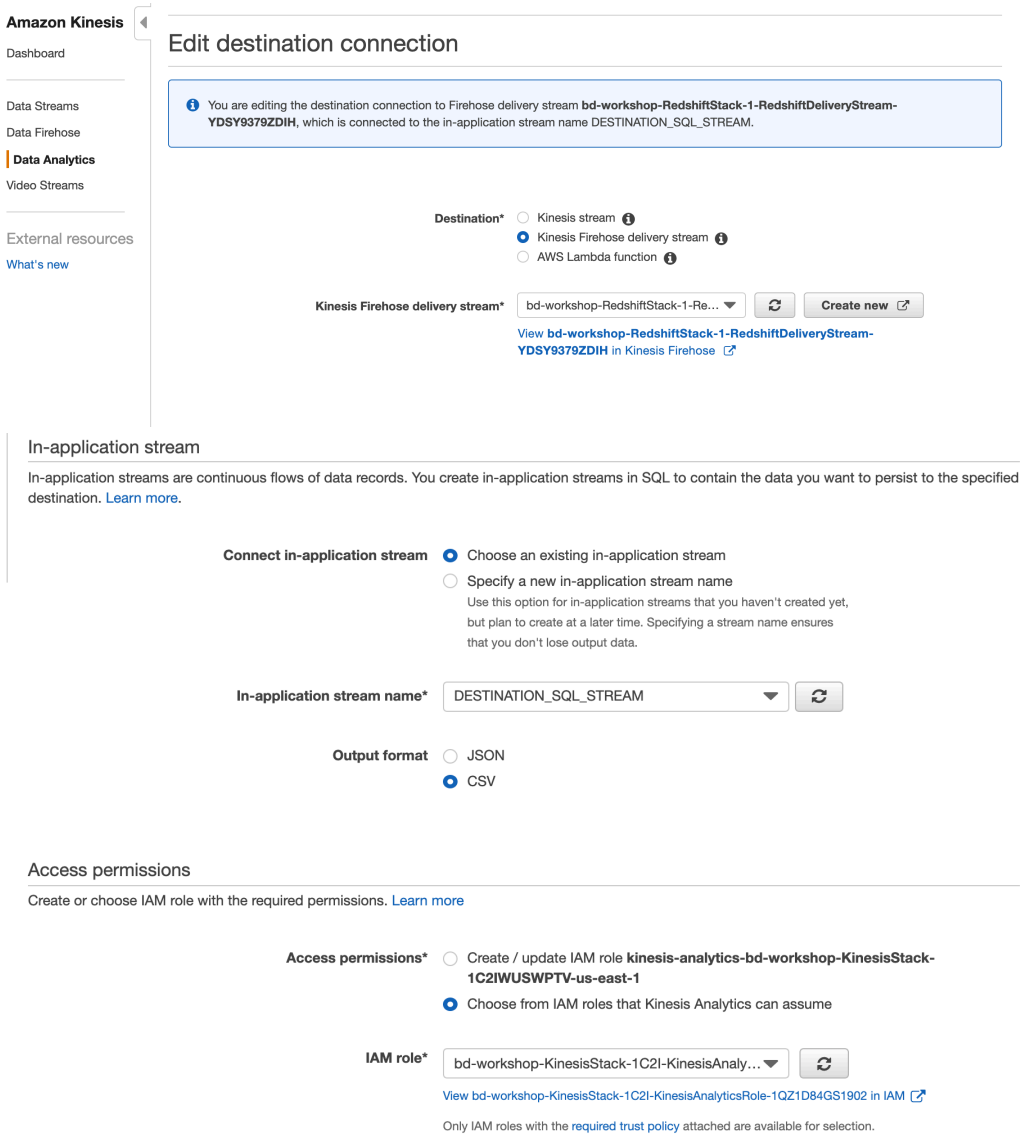
3.3.5.2 Keep the default for “Choose an existing in-application stream”.

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”



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;
```

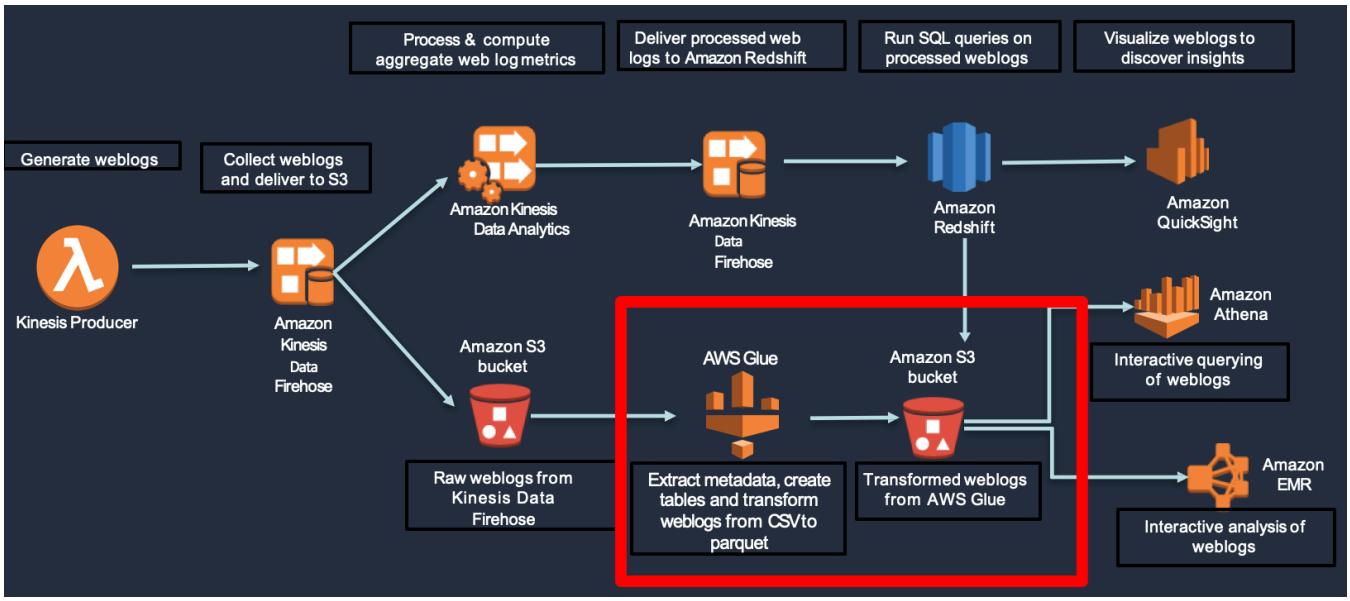
- Show all requests paths with status “PAGE NOT FOUND"

```
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:

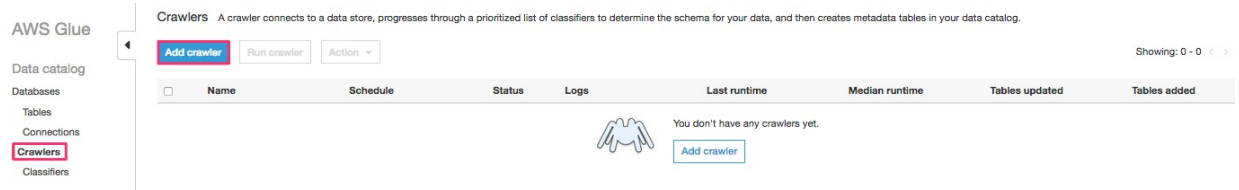
1. Discover and catalog the weblog data deposited into the Amazon S3 bucket using AWS Glue crawler
2. Transform weblogs to parquet format using the AWS Glue ETL job authoring tool



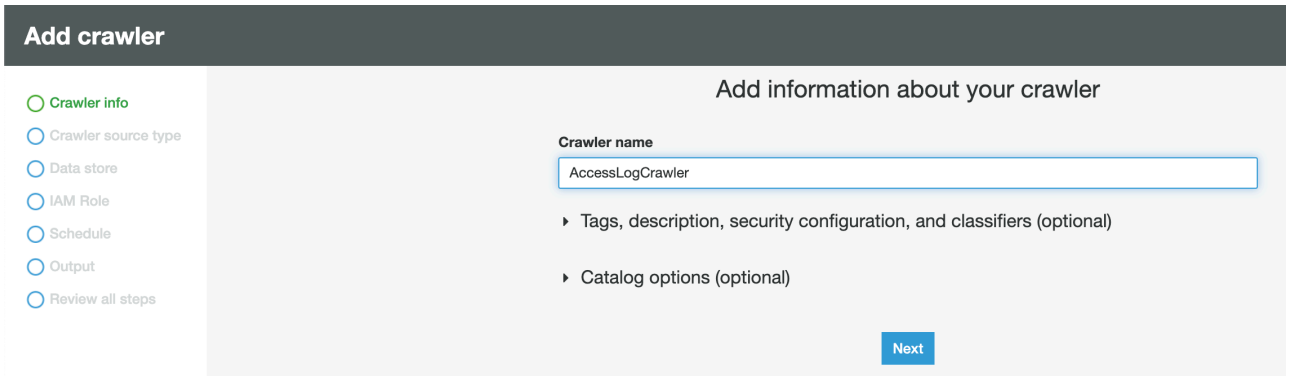
4.1 Discover dataset with AWS Glue

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

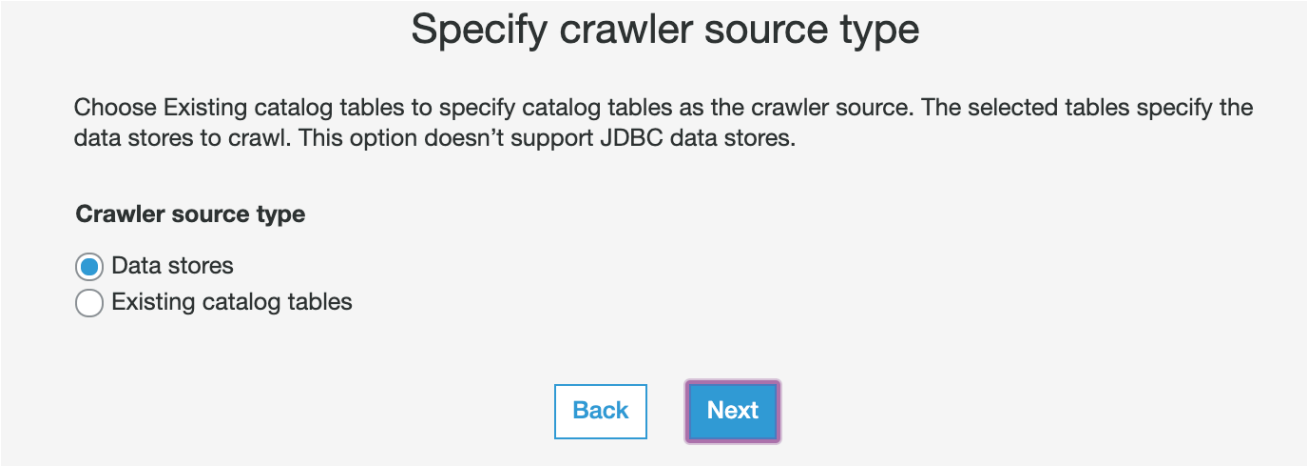
- From the AWS Management Console,
- Select / search **AWS Glue**.
- Click on "Get Started" on the next screen (if appears)
- Select "**Crawlers**" section on the left-hand side panel
- Click on "**Add crawler**"



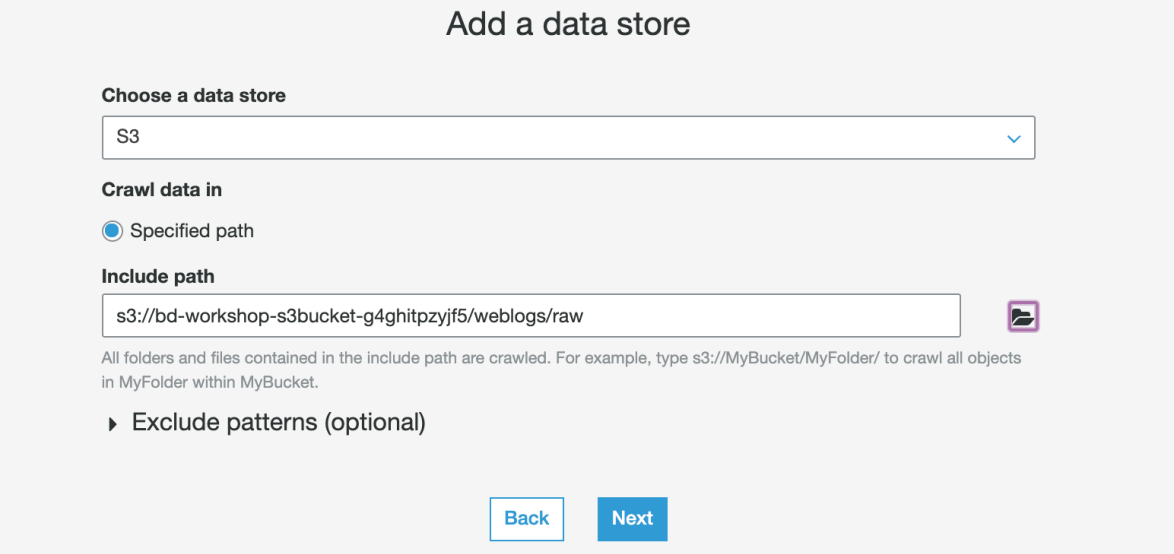
- Specify a name for the crawler. Click "**Next**"



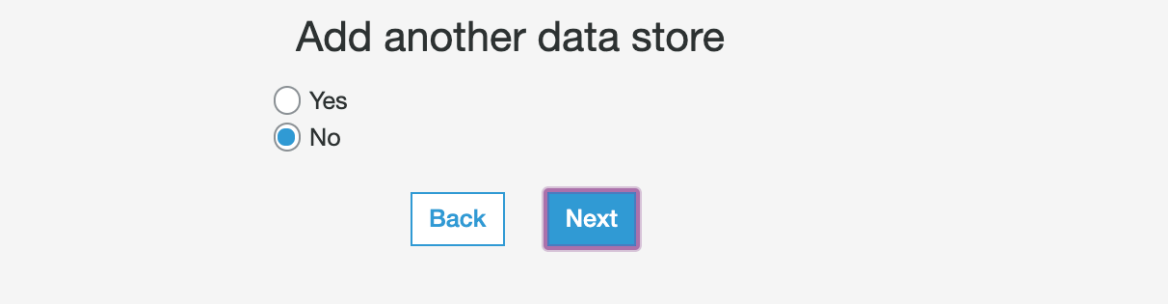
- Select Crawler source type to “Data Stores” and click "Next"



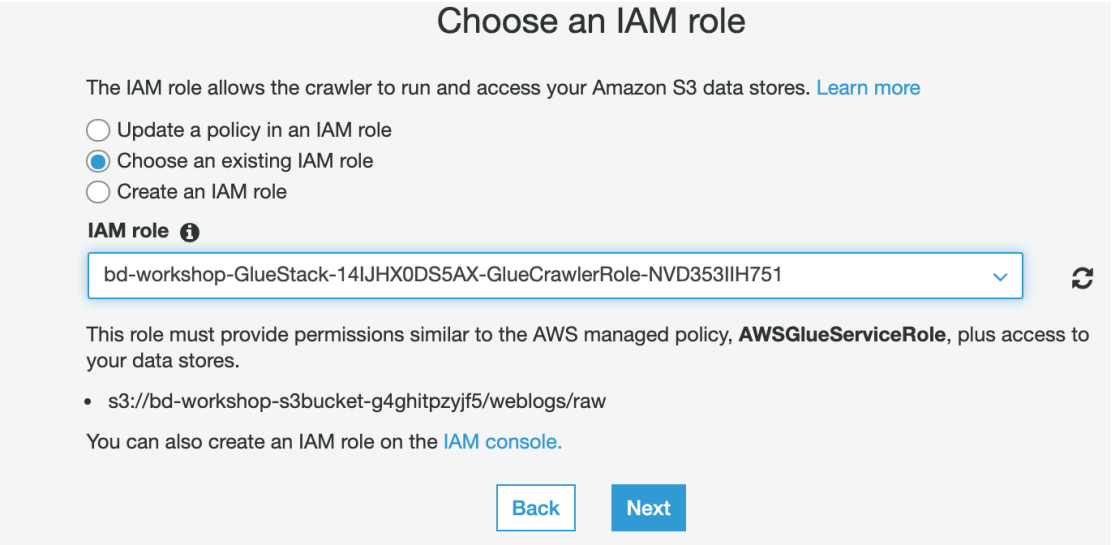
- Provide S3 path location where the raw weblogs were placed (navigate to S3 path: s3://<S3 bucket from template>/weblogs/raw)



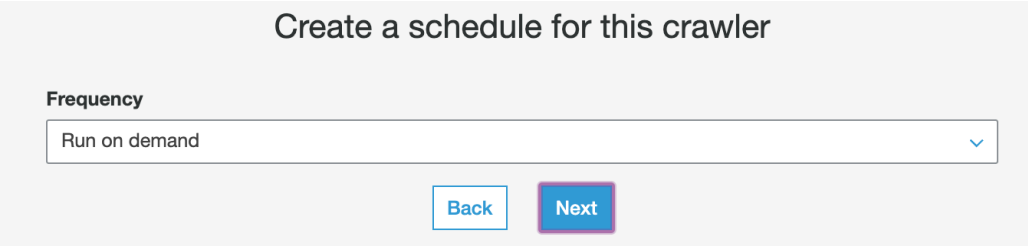
- Click "**Next**"
- Add Another data store – choose No.



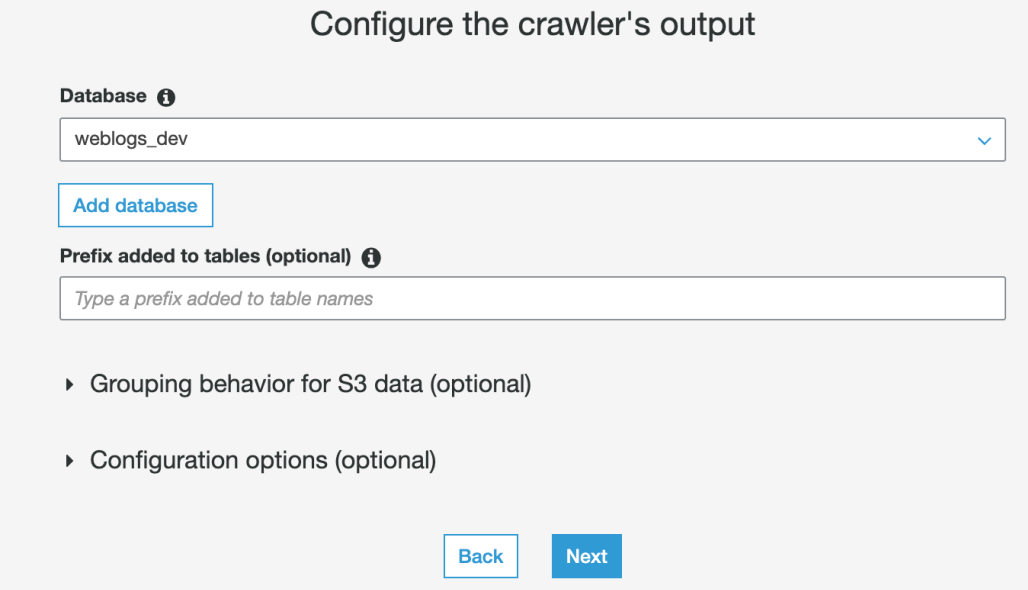
- Click "**Next**"
- In the IAM role section, select “Choose an existing IAM role”
- Select role <StackName>-GlueStack-xxx-GlueCrawlerRole-xxxxxxx as the IAM role



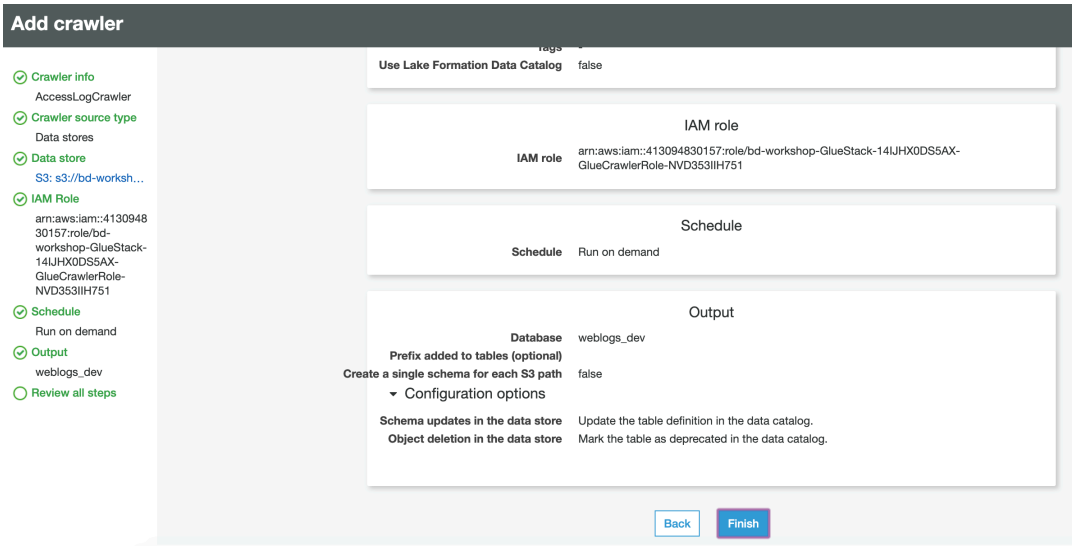
- Click "**Next**"
- Choose "**Run on demand**" to run the crawler now, and click "**Next**"



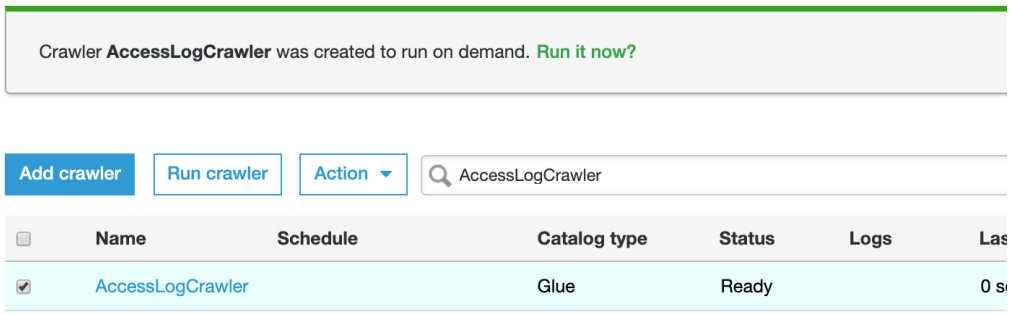
- On the next screen, drop down Database and select "**weblogs_dev**" and Click "Next"



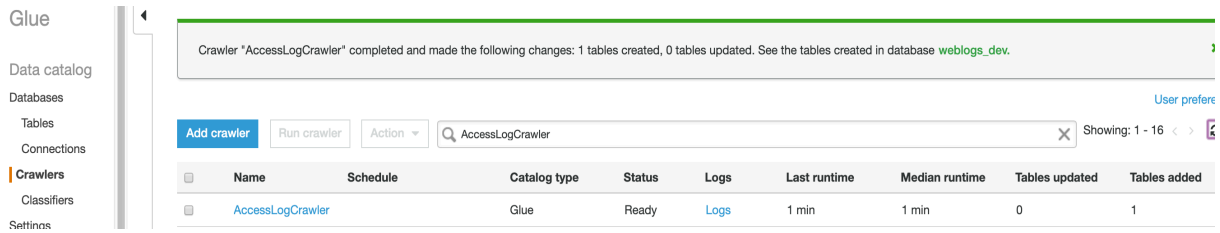
- Review and click "**Finish**" to create a crawler



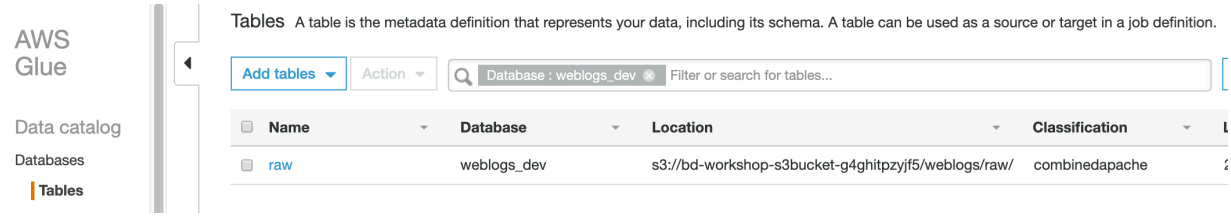
- Click on "**Run it now?**" link to run the crawler



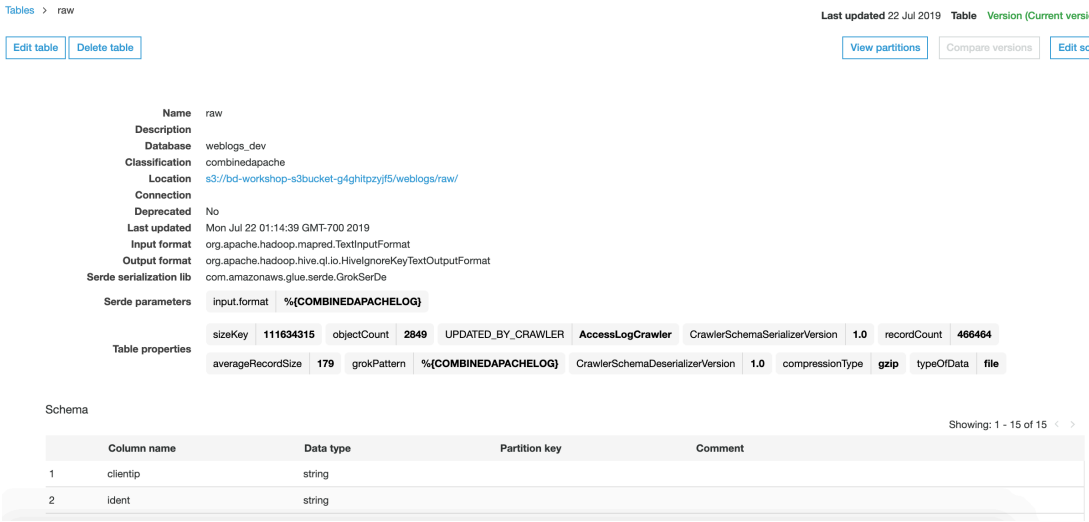
- Now wait for 1-2 min and Crawler shows a **Ready** status when it is finished running



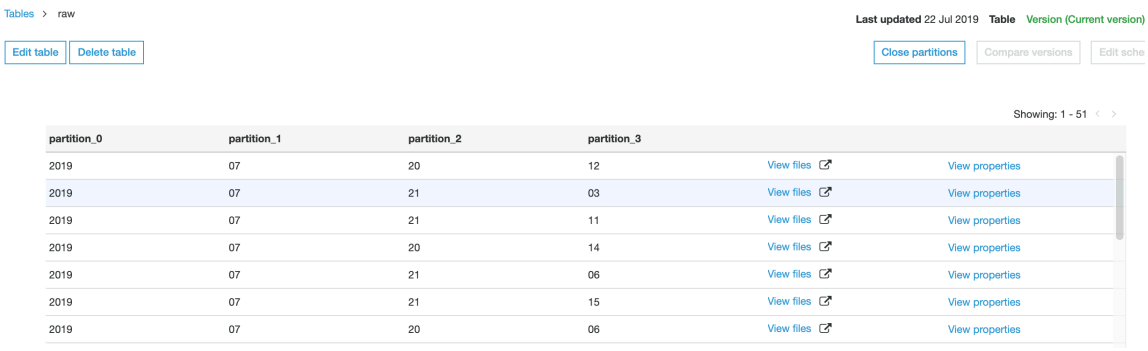
- Observe that the crawler has created a table for your dataset
- The crawler automatically classified the dataset as **combinedapache** log format



- Click the table to take a look at the properties

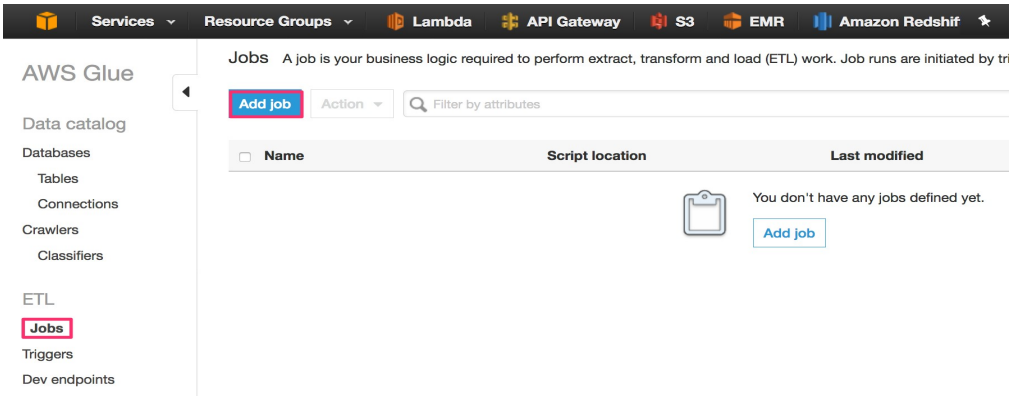


- AWS Glue used the **GrokSerDe** (Serializer/Deserializer) to correctly interpret the weblogs
- You can click on the "**View partitions**" link to look at the partitions in the dataset

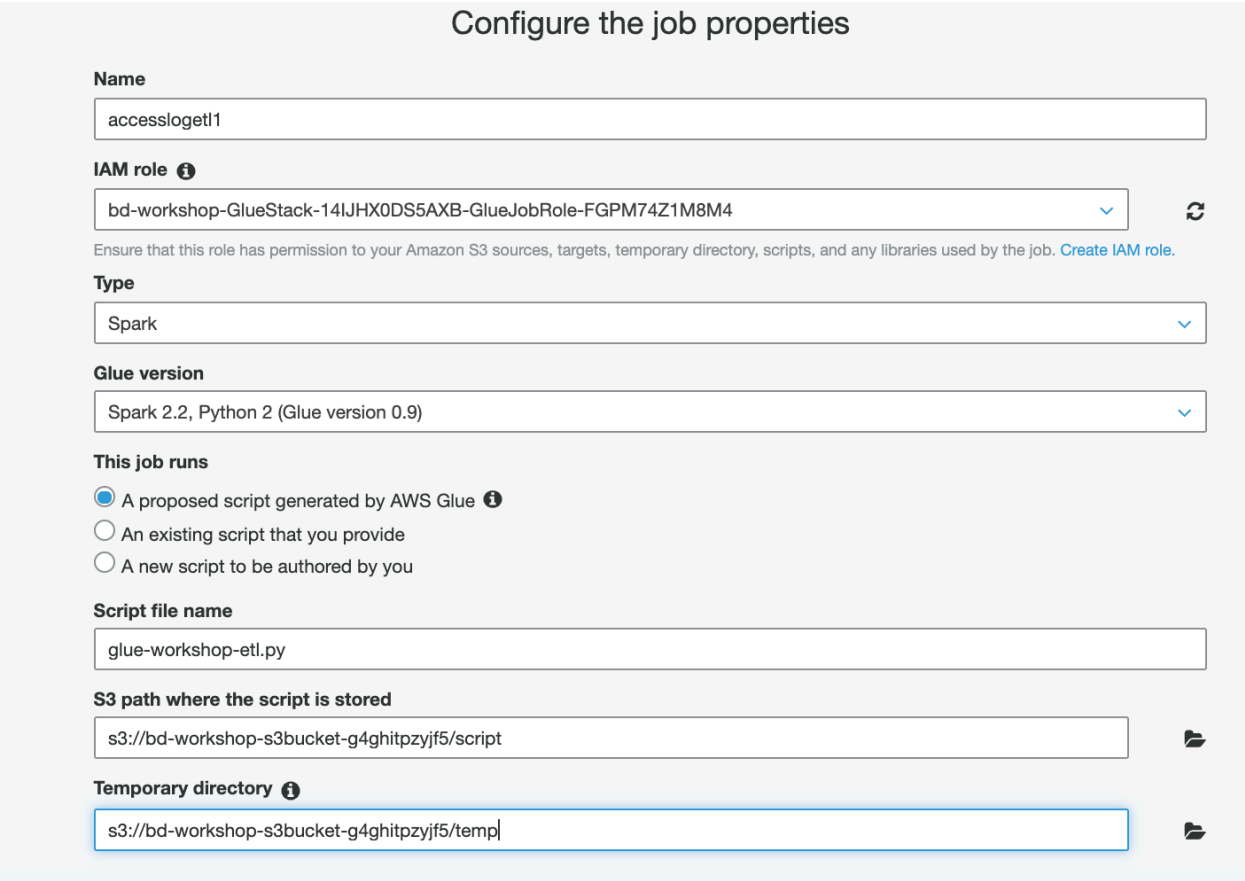


4.2 Create ETL job in AWS Glue

- 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.

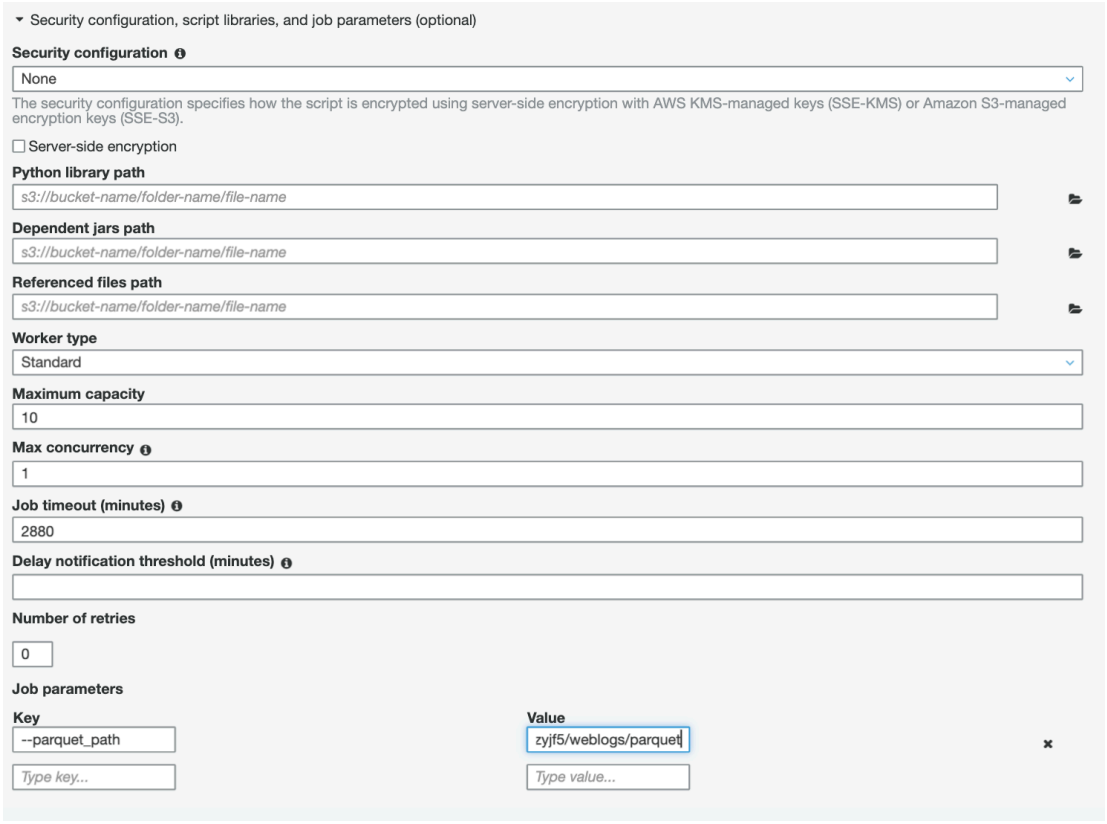


- Job name: **accesslogetl1**
- Select the IAM role <StackName>-GlueStack-xxx-GlueJobRole-xxxxxxx
- Select Type as Spark
- Select “a new script to be authored by you” Script file name: **glue-workshop-etl.py**
- For the S3 path where the script will be stored, use path **s3://<S3 bucket from template>/script**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

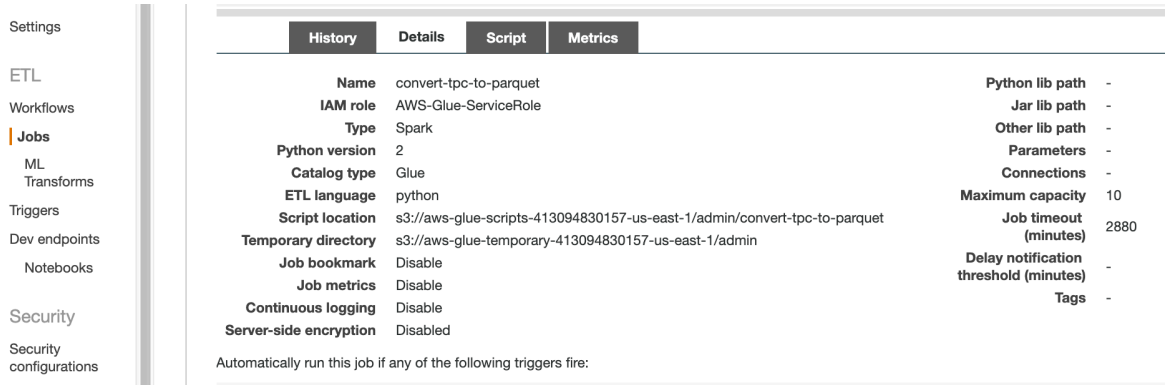


- **DO NOT CLICK NEXT JUST YET**

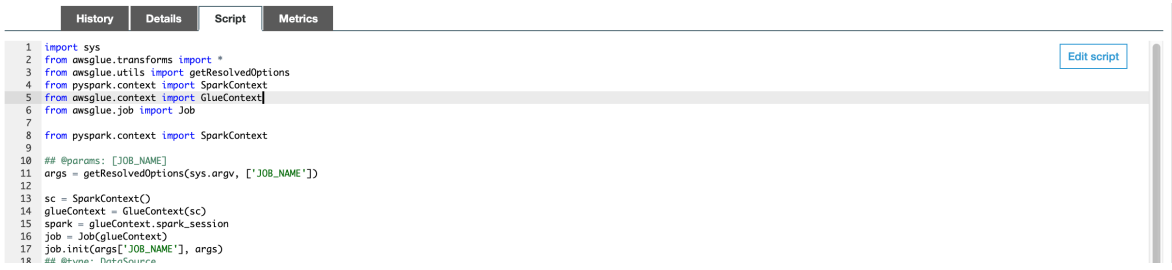
- Expand **Security configuration, script libraries, and job parameters (optional)** section,
    - Set the Maximum capacity (DPUs) to 10.
    - Let's pass a job parameter to send the S3 path where parquet files will be deposited.
    - Specify the following values for Key and Value
    - **Key: --parquet_path** (notice the 2 hyphens at the beginning and underscore between "parquet" and "path")
    - **Value: s3://<S3 bucket from template>/weblogs/parquet**



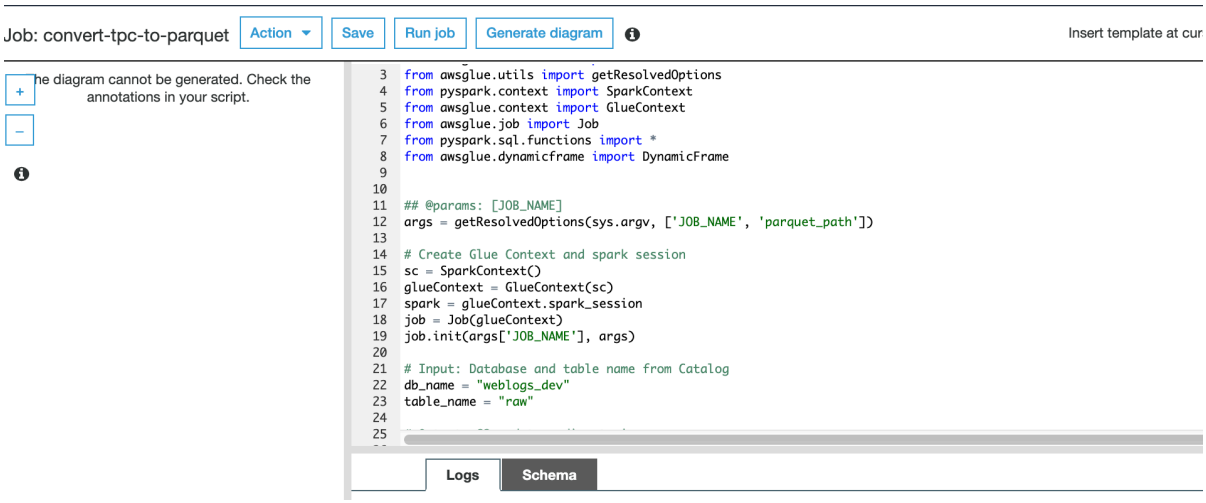
- Click "**Next**" in the following screen
- Choose any source path and **Next**
- Choose any target path and **Next**
- Review and click "**Save Job**" to create the job



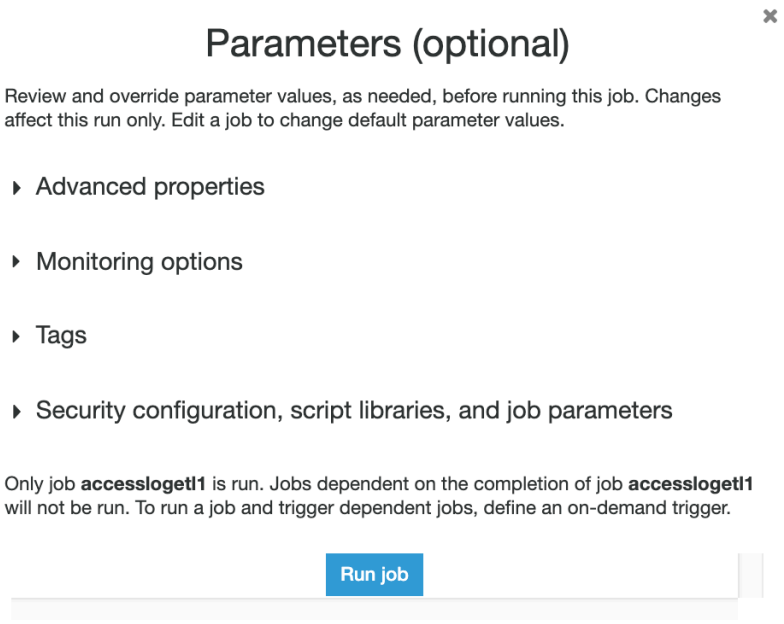
- Select the ETL job “accesslogetl1” and click on “**Edit Script**”.

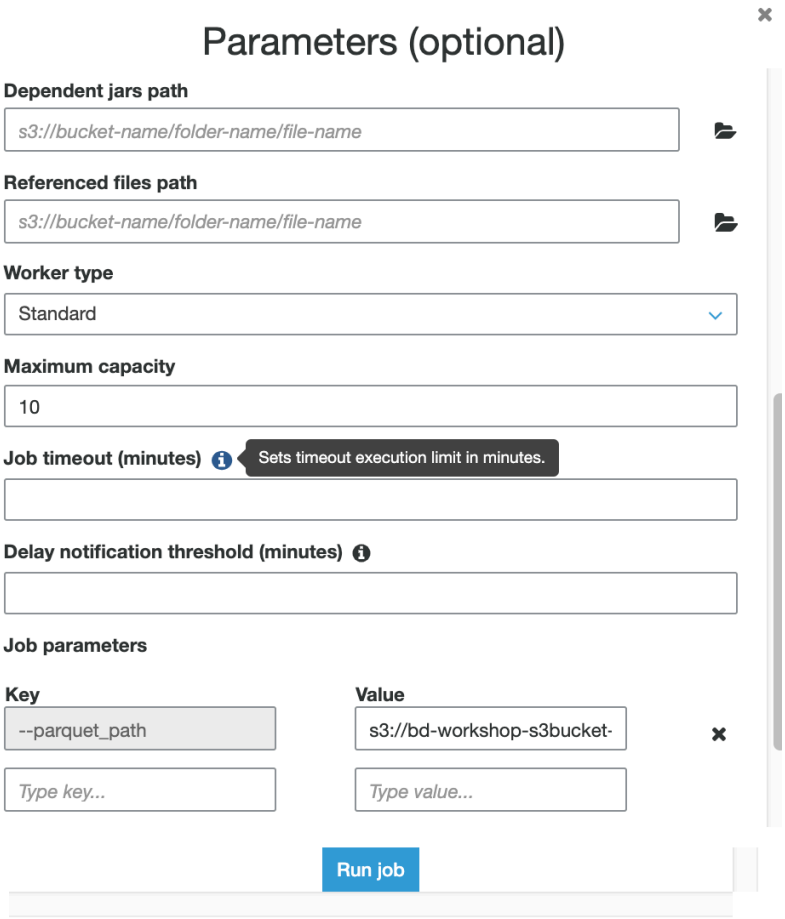


- Copy the ETL code in glue-workshop-etl.py from BigDataWorkshop.zip and paste into ETL job editor. (Overwrite; don't append.)



- 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**





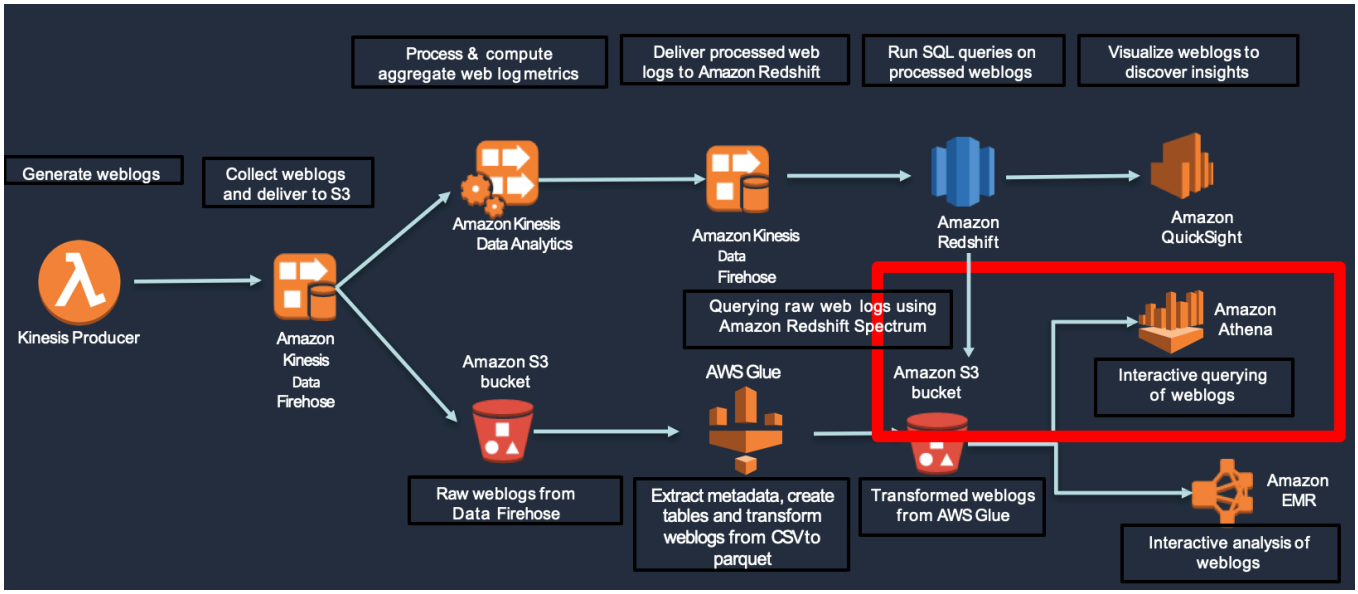
- Click "**Run job**" to continue.
  Note: This might take a few minutes. When the job finishes, weblogs will be transformed to parquet format.

- 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)

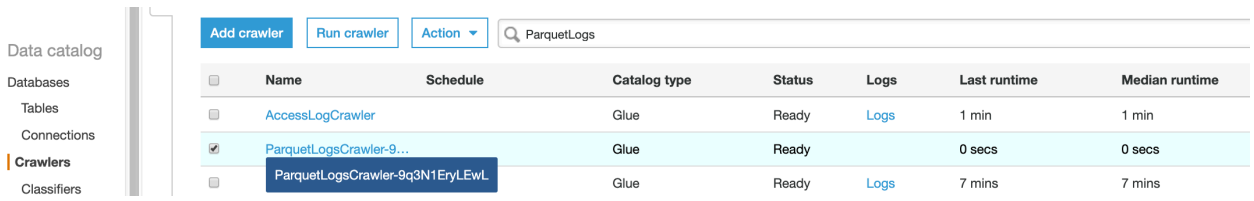
We are going to:

A. 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.
B. Run queries from Amazon Redshift on the parquet weblogs in Amazon S3 using Amazon Redshift Spectrum.
C. Run interactive queries from Athena on parquet weblogs in S3.

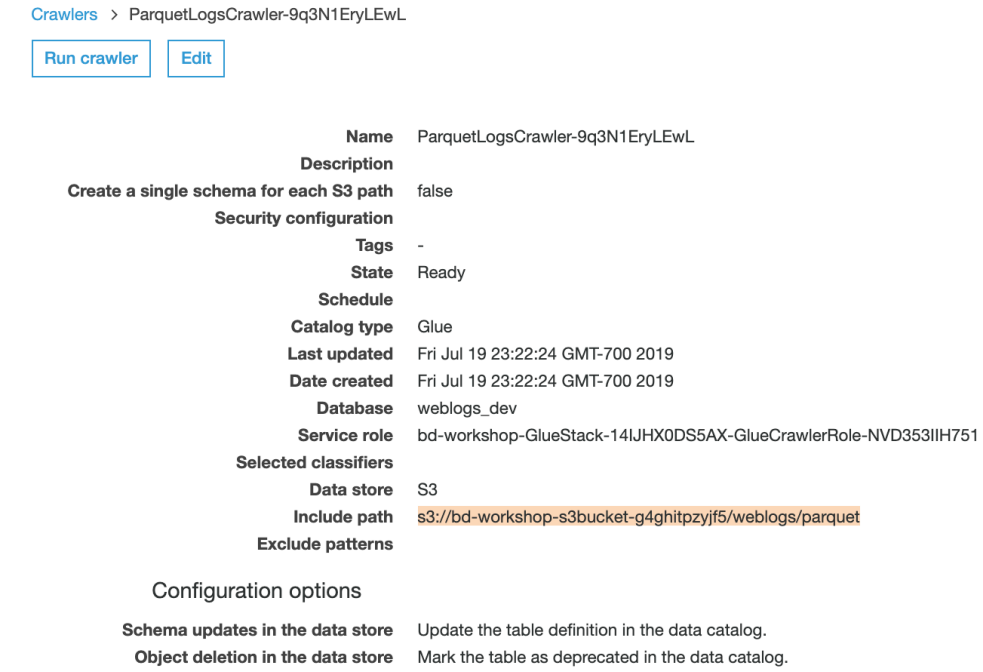


5.1 Set up AWS Glue crawler for processed parquet data

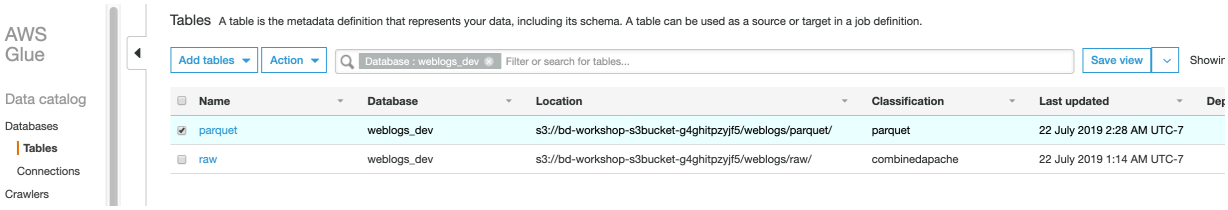
- Go to Crawlers in AWS Glue and search for **ParquetLogs**.
- Select the ParquetLogsCrawler-xxxxxxx.



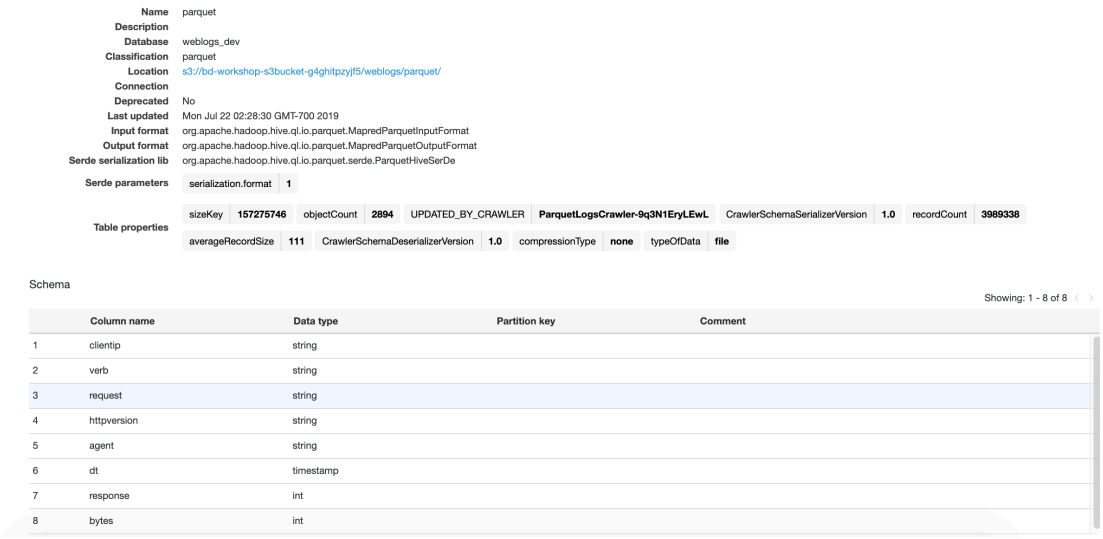
- Go to details tab and check that it's pointing to the right parquet location in Amazon S3.



- 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.



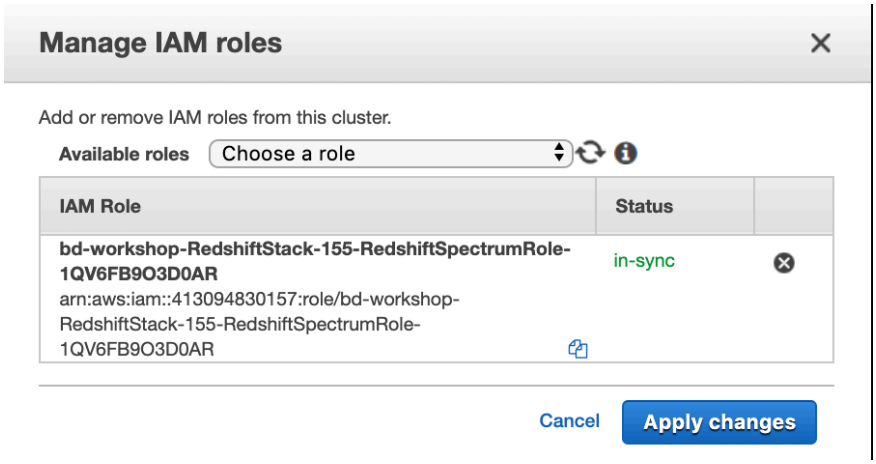
- Take a look at the schema of the parquet table



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-xxxxxxxxxxx.



- 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
```

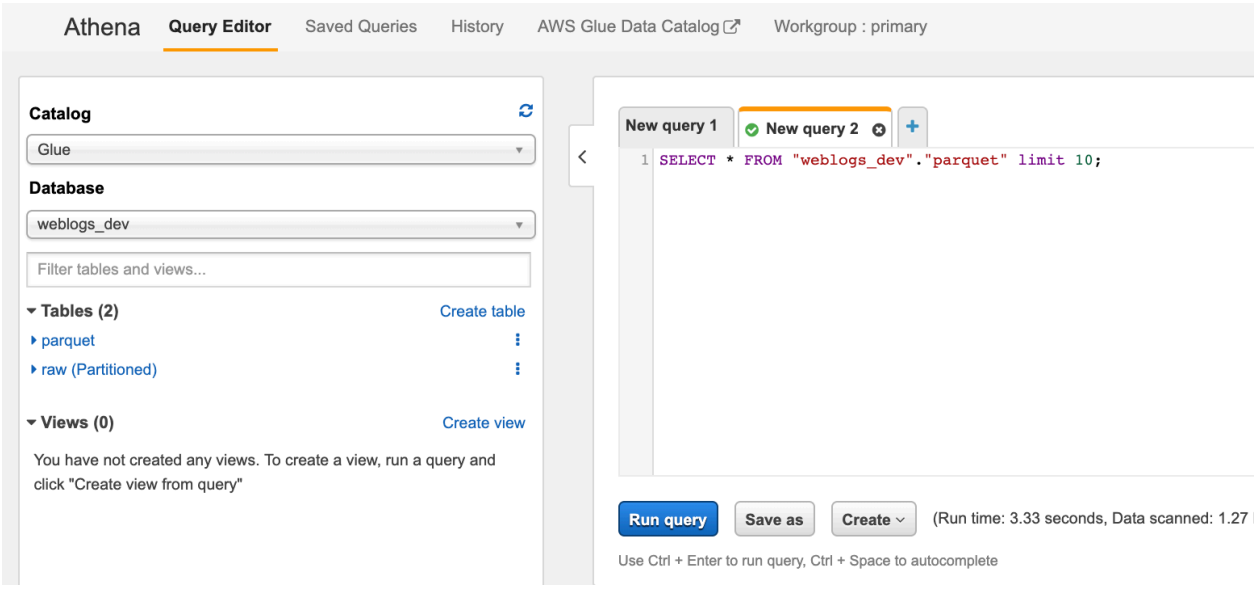
- Count of all records in the parquet location in Amazon S3 where the response is 404

```
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

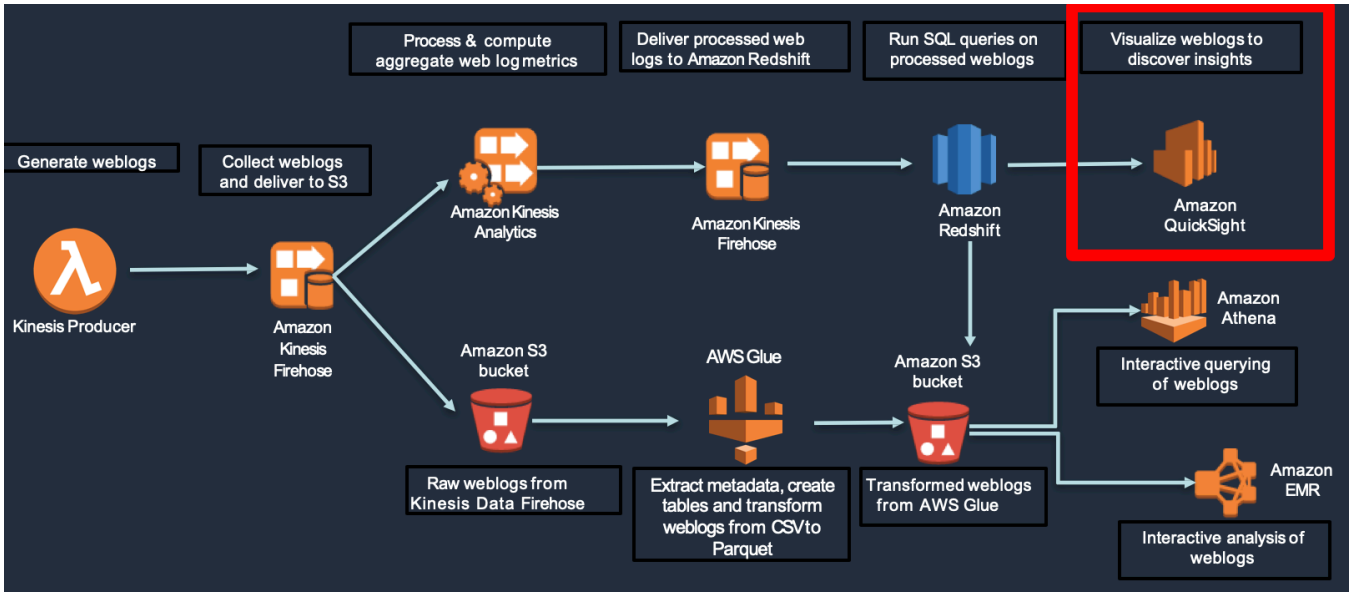
```
-- Access Log Queries
SELECT clientip, COUNT(1) as cnt FROM weblogs_dev.parquet GROUP BY 1;

-- 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;
```



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

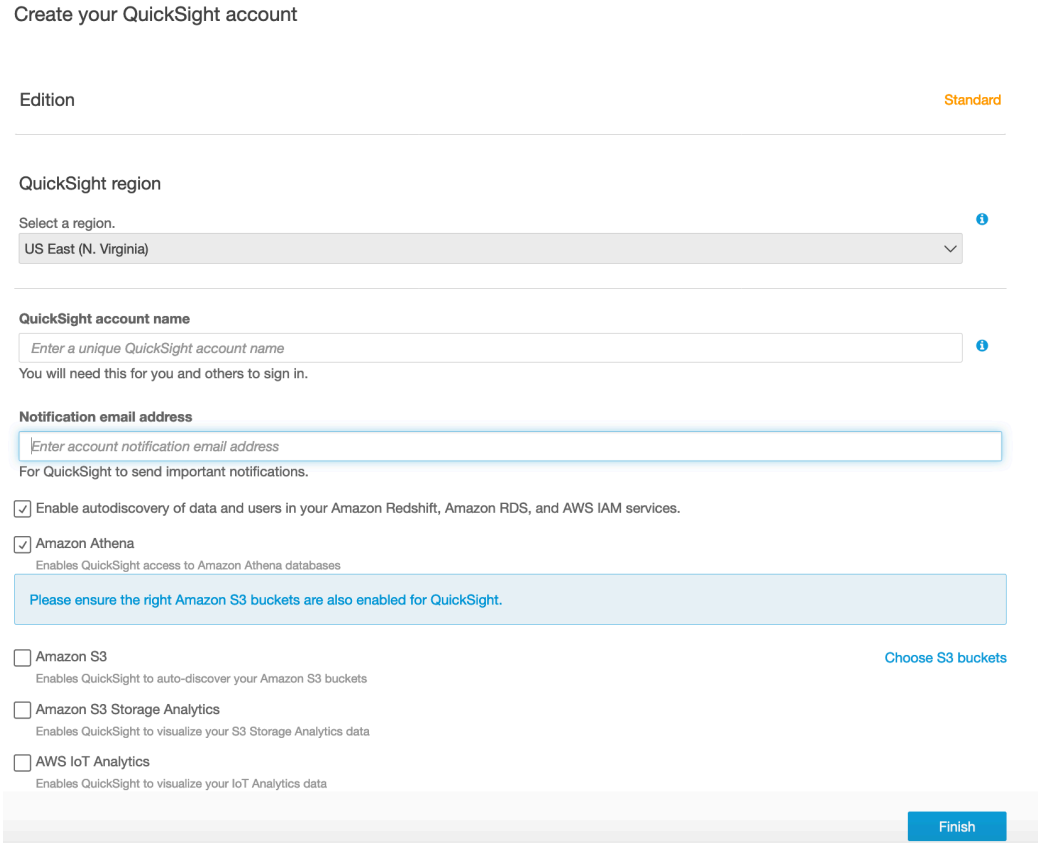
Your AWS Account is not signed up for QuickSight. Would you like to sign up now?

AWS Account 0[redacted]7

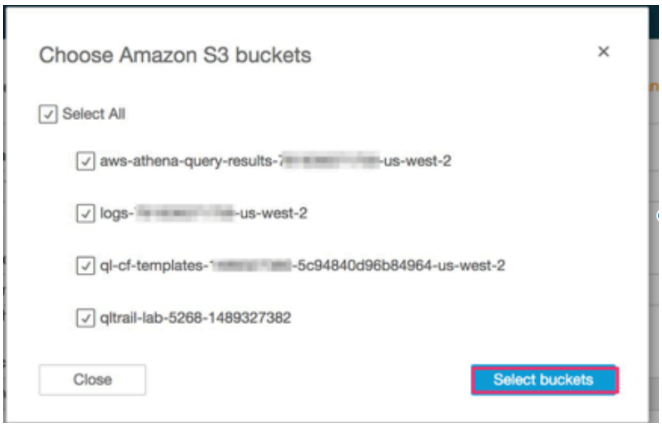
Sign up for QuickSight

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



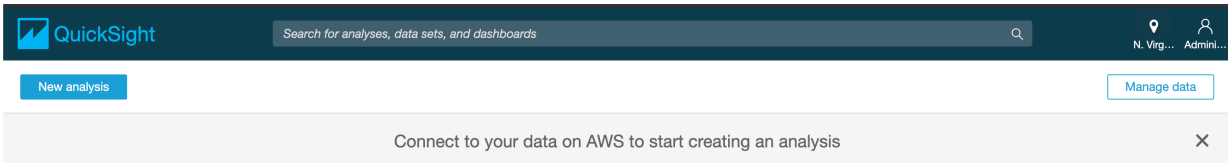
- On the "Subscription type" page, enter the account name (see **note** below)
  Note: Amazon QuickSight account name is your AWS account number
- Enter your **email address**
- Select **< AWS region>** where you created CloudFormation stack.
- Check the "**Amazon S3 (all buckets)**" box



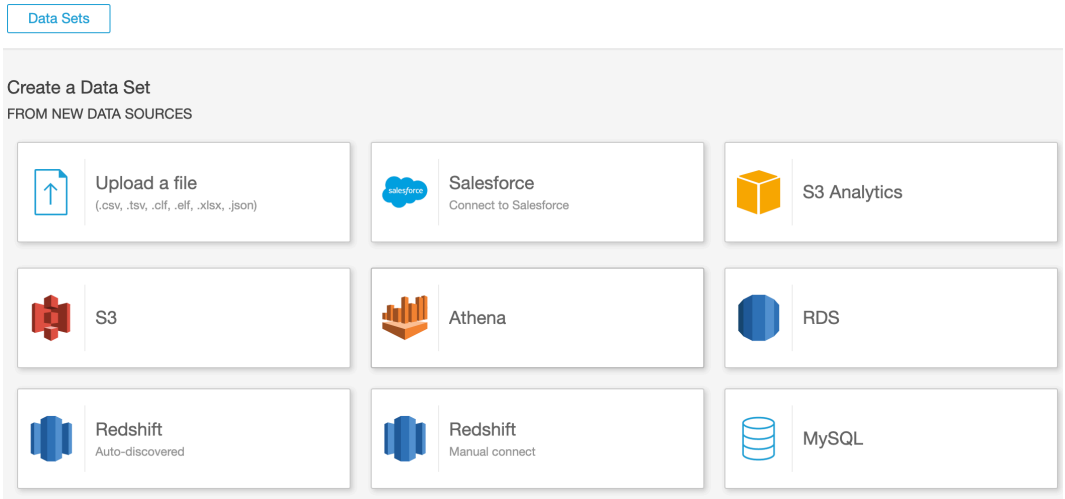
- Click “Select buckets”
- Click “Finish”.

6.2 Connect to Amazon Redshift (Data Store) for QuickSight

- Click on "**Manage Data**"



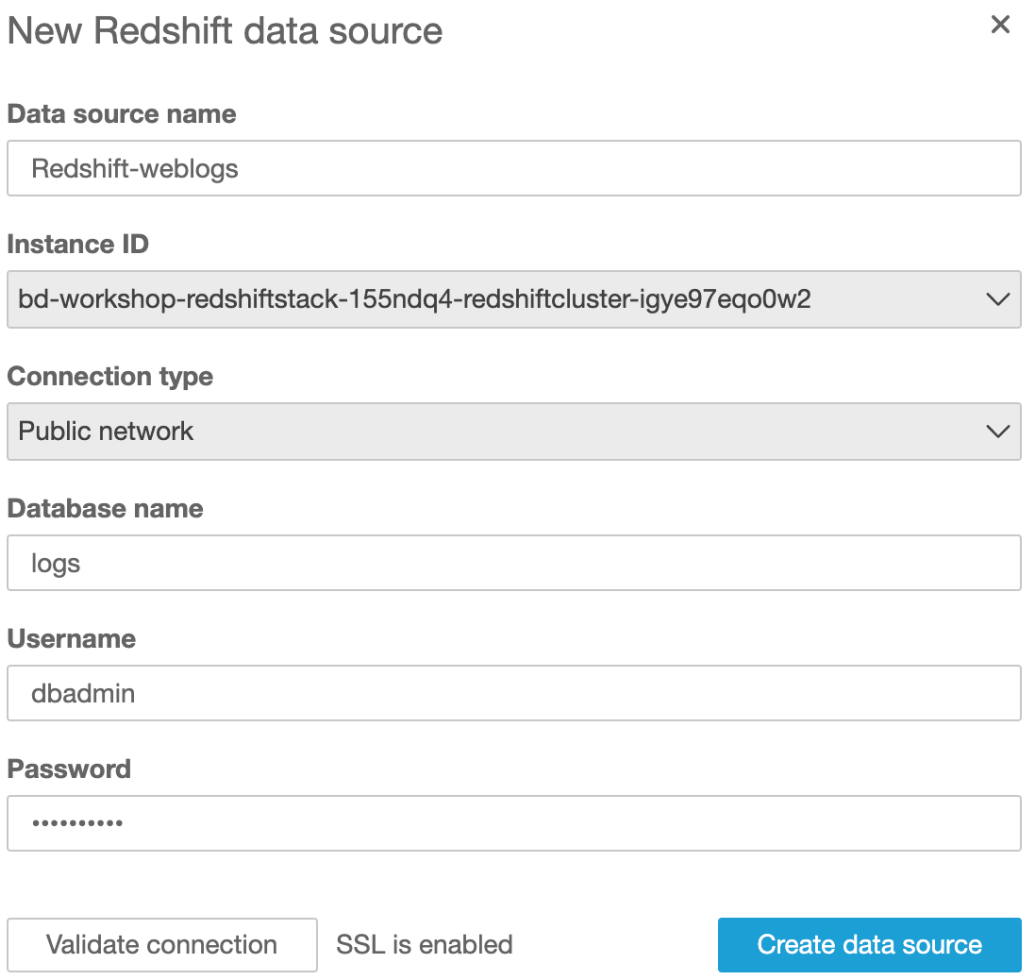
- select "**New Data set**" to create a new data set in Amazon QuickSight



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

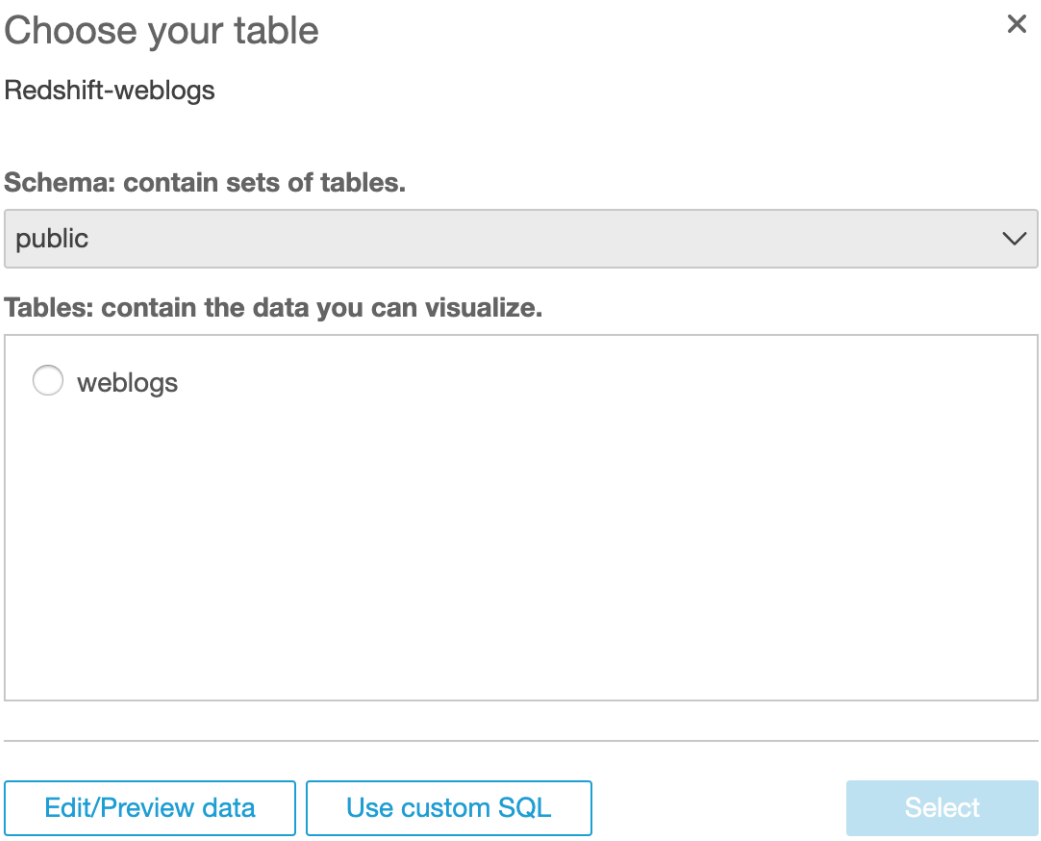
  Note: Amazon QuickSight auto-discovers databases associated with your AWS account (Amazon Redshift database in this case).

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

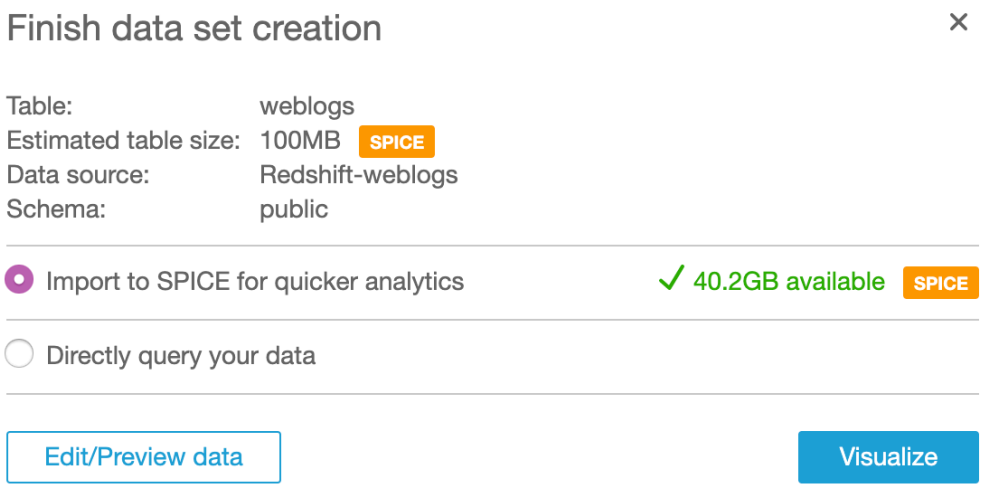


- Click on “Validate connection”. It will change status to Validated if entered details are correct.

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



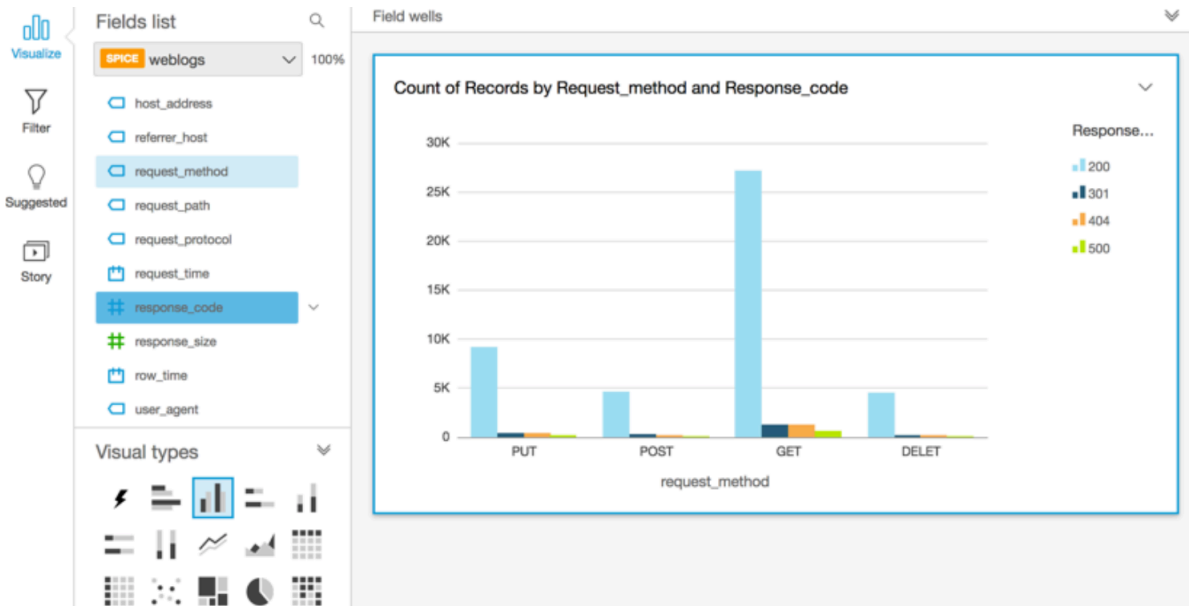
- Click “Select”.
- Choose - Import to SPICE for quicker analytics



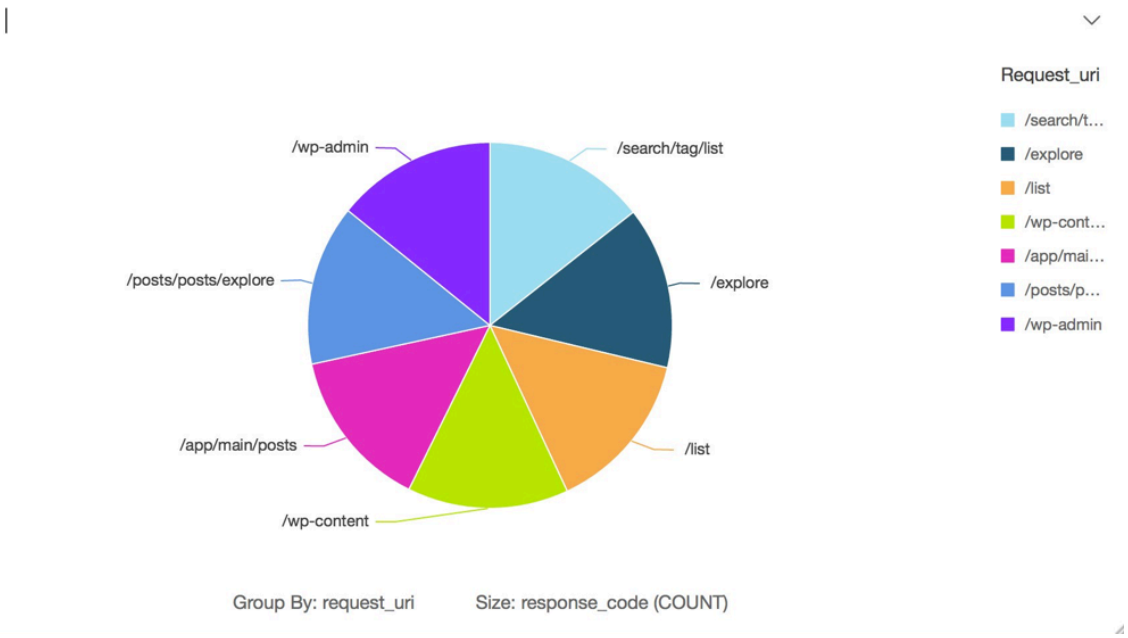
- Click “Visualize”.

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



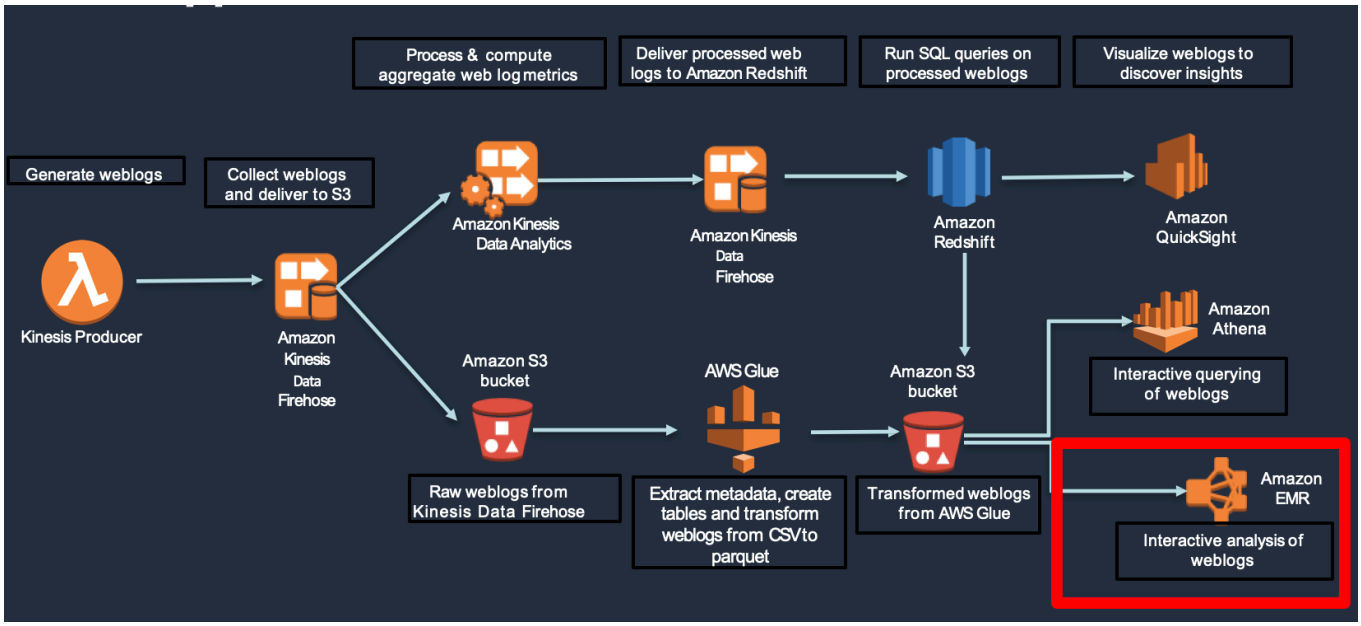
- Add a visual to demonstrate which URI are the most requested



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

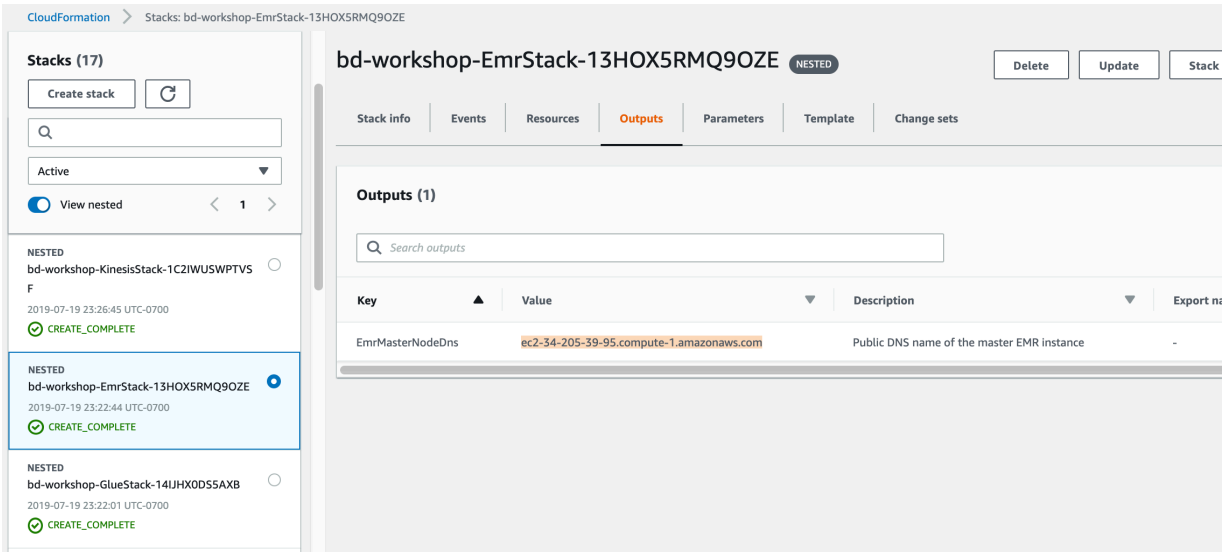
We are going to:

- 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

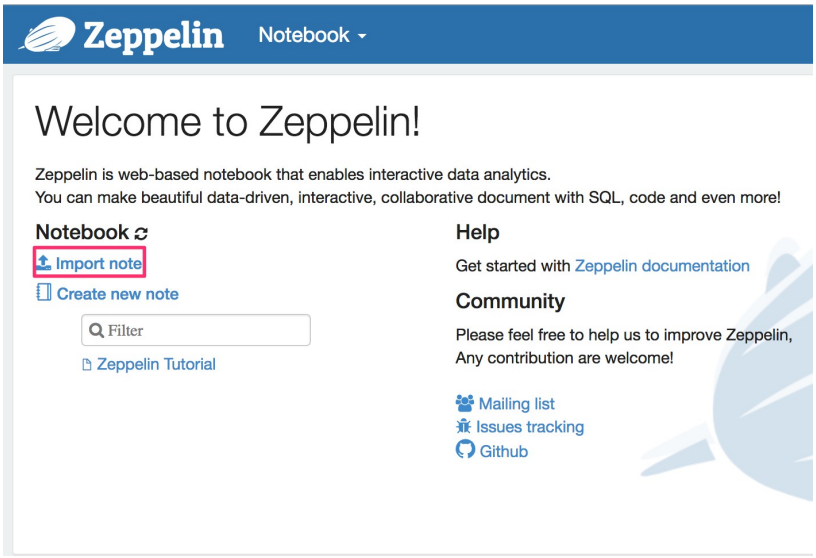


7.1 Open the Zeppelin interface

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

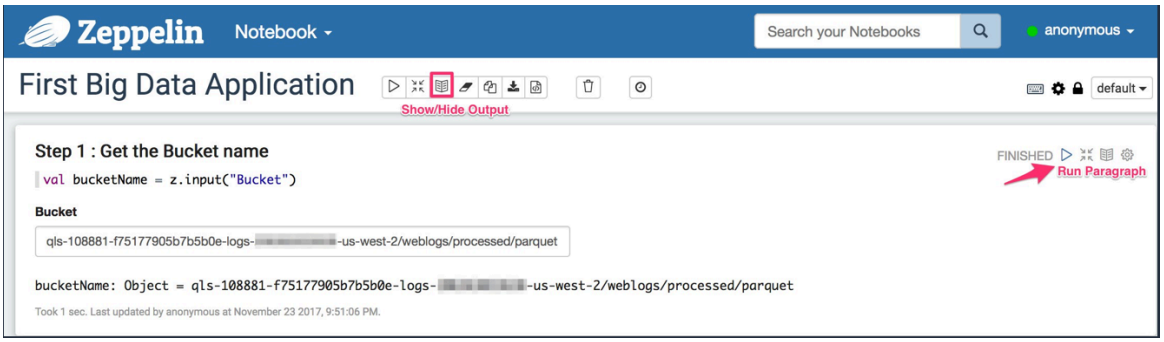


- Open the Zeppelin link in a new browser window



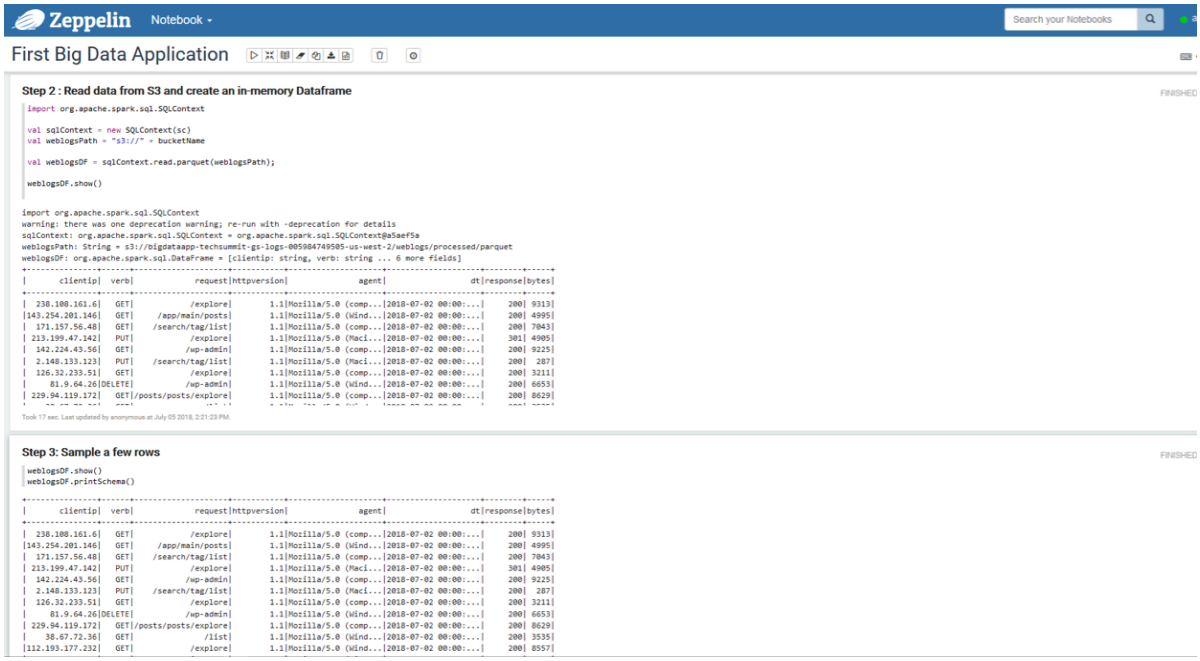
- Open the First Big Data Application json file from the BigDataWorkshop.zip
- Import the notebook using the "Import Note" link on Zeppelin interface
  Note: You may need to disconnect from VPN or the page will not load

- 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

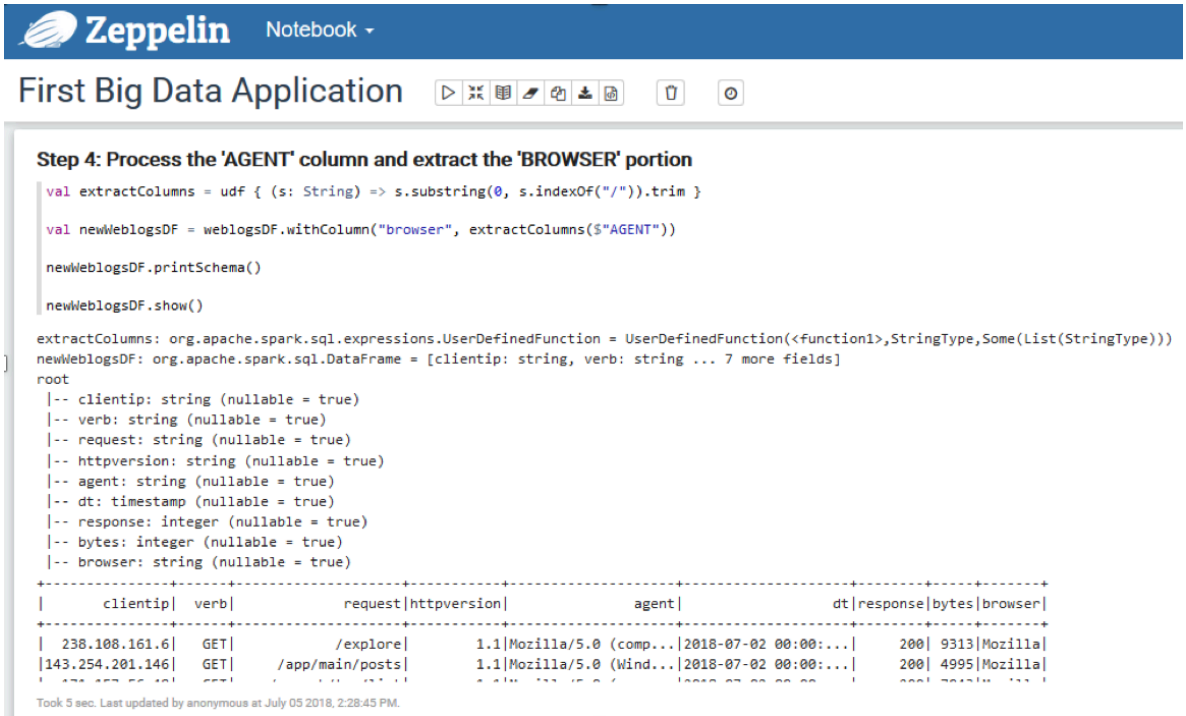


7.2 Run the notebook

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



- 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



- 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
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.