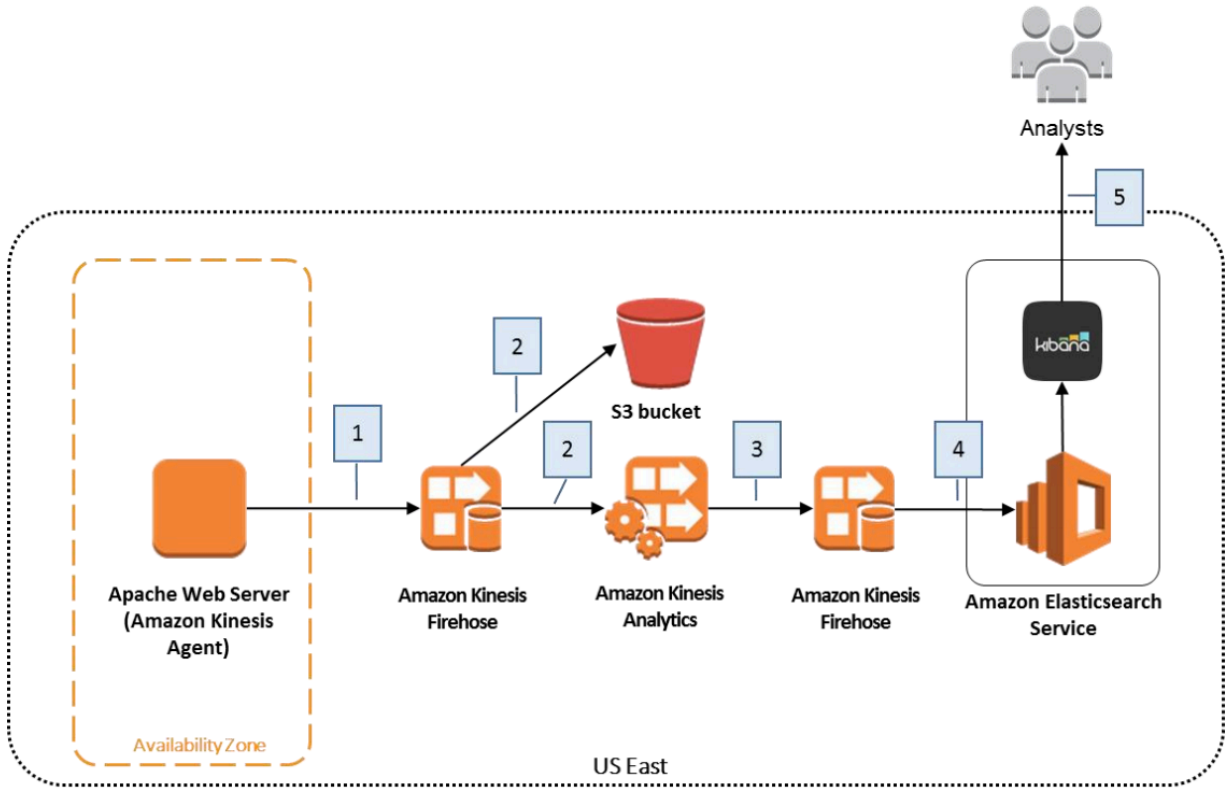
Web Log Analytics Solution on AWS

Amazon Kinesis Analytics is a very convenient way to process streaming data in real time with Standard SQL. It enables creating and executing SQL queries on real-time, incoming streaming data so as to gain actionable insights and respond to business needs promptly.

In the tasks completed as part of this tutorial, I learnt how to ingest streaming log data, aggregate that stream data and persist the aggregated data so that same can be analyzed and visualized. Here I have created a complete end-to-end system that integrates several AWS services. I have analyzed a live stream of Apache access log data and aggregated the total request for each HTTP response type every minute. To visualize the results in near real-time, I have used a UI Tool, Kibana to chart the results.

One major benefit of using these AWS services for log analysis, was that the entire analysis infrastructure was created with a serverless architecture. The system was created by integrating **Amazon Kinesis Firehose, Amazon Kinesis Analytics, Amazon Elasticsearch Service**. The architecture of the solution can be seen below –



[1]
The Web Server used for log generation, is an Amazon EC2 instance. I installed an Amazon Kinesis Agent on this Linux instance to continuously forward log data to an Amazon Kinesis Firehose delivery stream.

[2]
Amazon Kinesis Firehose is connected to two processes: one to write each log record to an Amazon S3 bucket (for durable storage of the raw log data), and another to an Amazon Kinesis Analytics application, for providing stream data as input to the application for analysis.

[3]
The Amazon Kinesis Analytics application is continuously running an SQL statement against the streaming input data. It creates an aggregated dataset every minute, and then outputs that data to a 2nd Firehose delivery stream.

[4]
The 2nd Firehose delivery stream writes the aggregated data to an Amazon ES domain.

[5]
Finally, a view of the streaming data is created using Kibana, to visualize the system output.

Step 1: Starting EC2 instance

I have used a **t2.micro** instance of the Operating System – **Amazon Linux AMI**

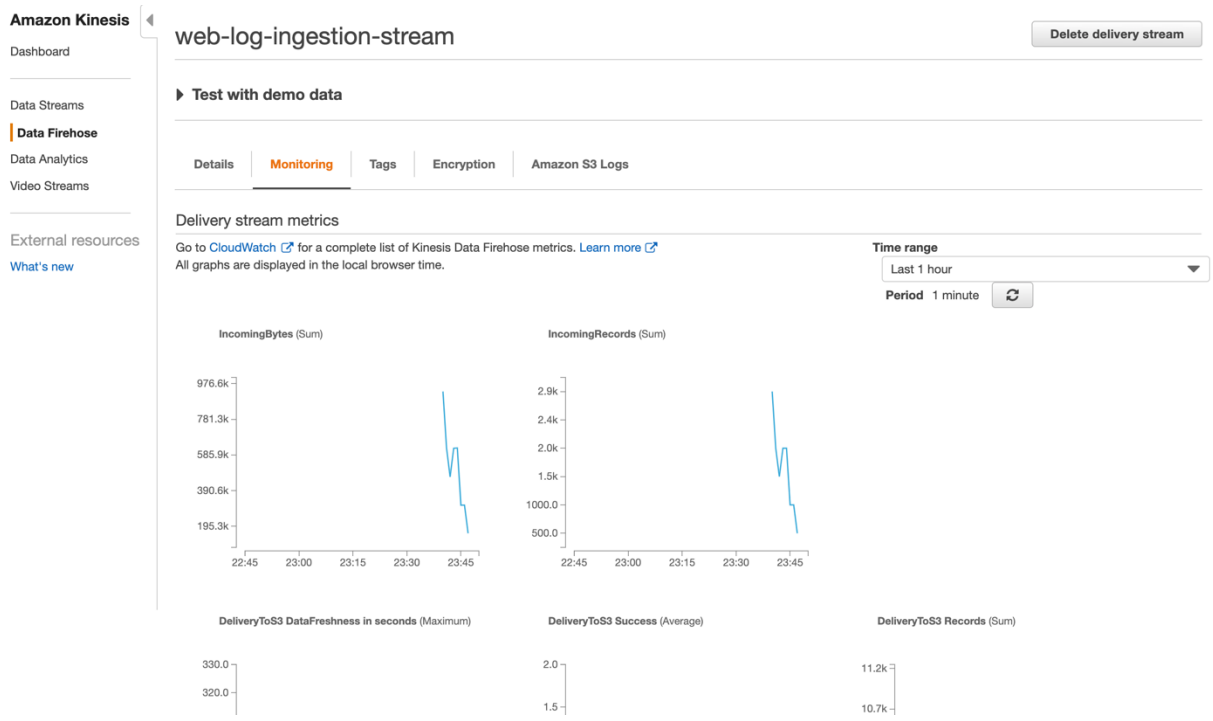
It is necessary for the instance to have an IAM role configured with permission to write to Amazon Kinesis Firehose and Amazon CloudWatch (for observing logs and events)

Step 2: Preparing Log Files

For generation of logs, and simulating a stream-like, real-time generation, I have downloaded, installed and executed the Fake Apache Log Generator from Github.

Step 3: Creating an Amazon Kinesis Firehose delivery steam

Before the created log files can be analyzed, they need to be loaded into some analysis service in AWS. Amazon Kinesis Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon S3, Amazon Redshift or Amazon Elasticsearch Service. I have used Firehose's delivery stream to propagate the real-time stream.



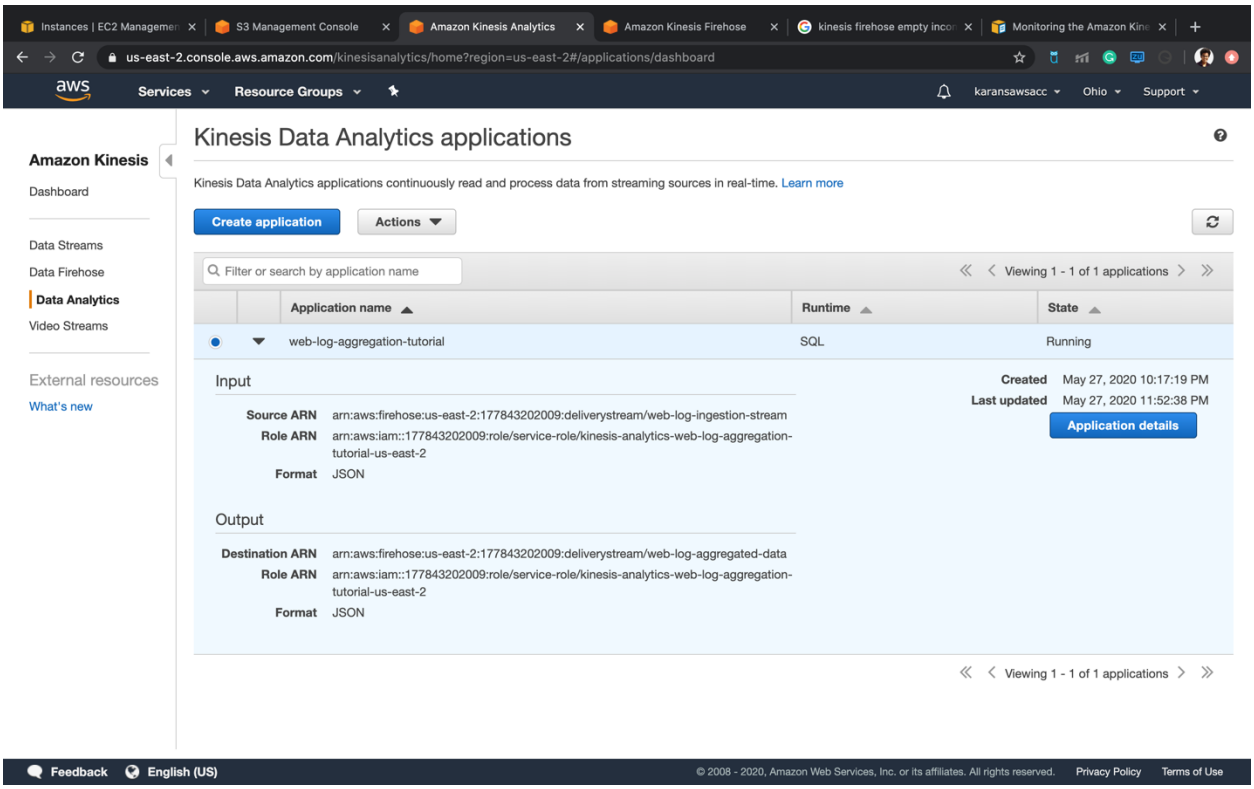
Step 4: Installing and Configuring Amazon Kinesis Agent

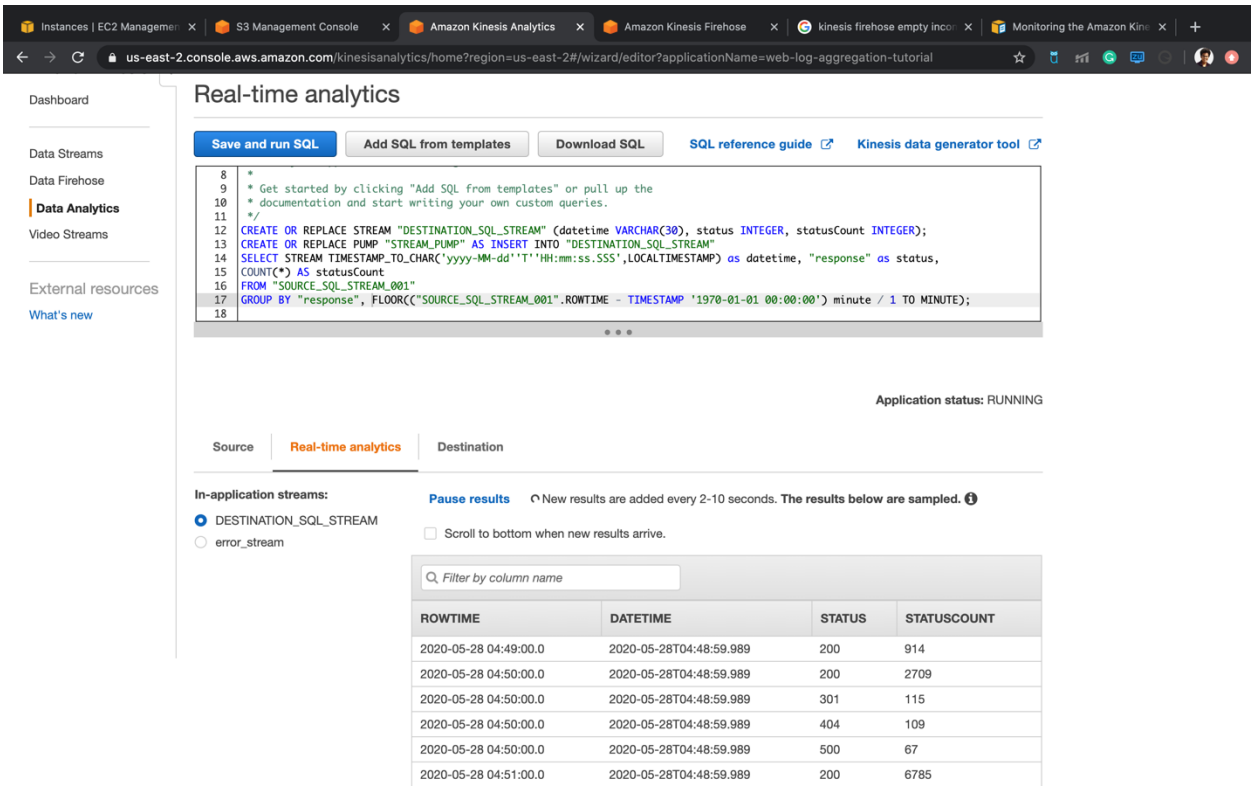
After readying the stream transmission service, to initiate the stream from the EC2 instance I installed an Amazon Kinesis agent software on the instance. This agent is a standalone Java application that offers an easy way to collect and send data to Firehose. This agent continuously monitors a set of files and sends new data to the delivery stream. Delivering all the data in a reliable, timely and simple manner, it also emits Amazon CloudWatch metrics to help in better monitoring and troubleshooting of the streaming process.
The agent is configured to parse log files in the Apache Common Log format and convert each line in the file to JSON format before sending it to the Firehose delivery stream.

Step 5: Creating an Amazon Elasticsearch Service
Data produced after loading in the first Firehose delivery stream, is stored in Amazon Elasticsearch (Amazon ES) for later visualization and analysis.

Step 6: Creating a second Amazon Kinesis Firehose delivery stream
For this second delivery stream, our destination will be Amazon Elasticsearch Service.

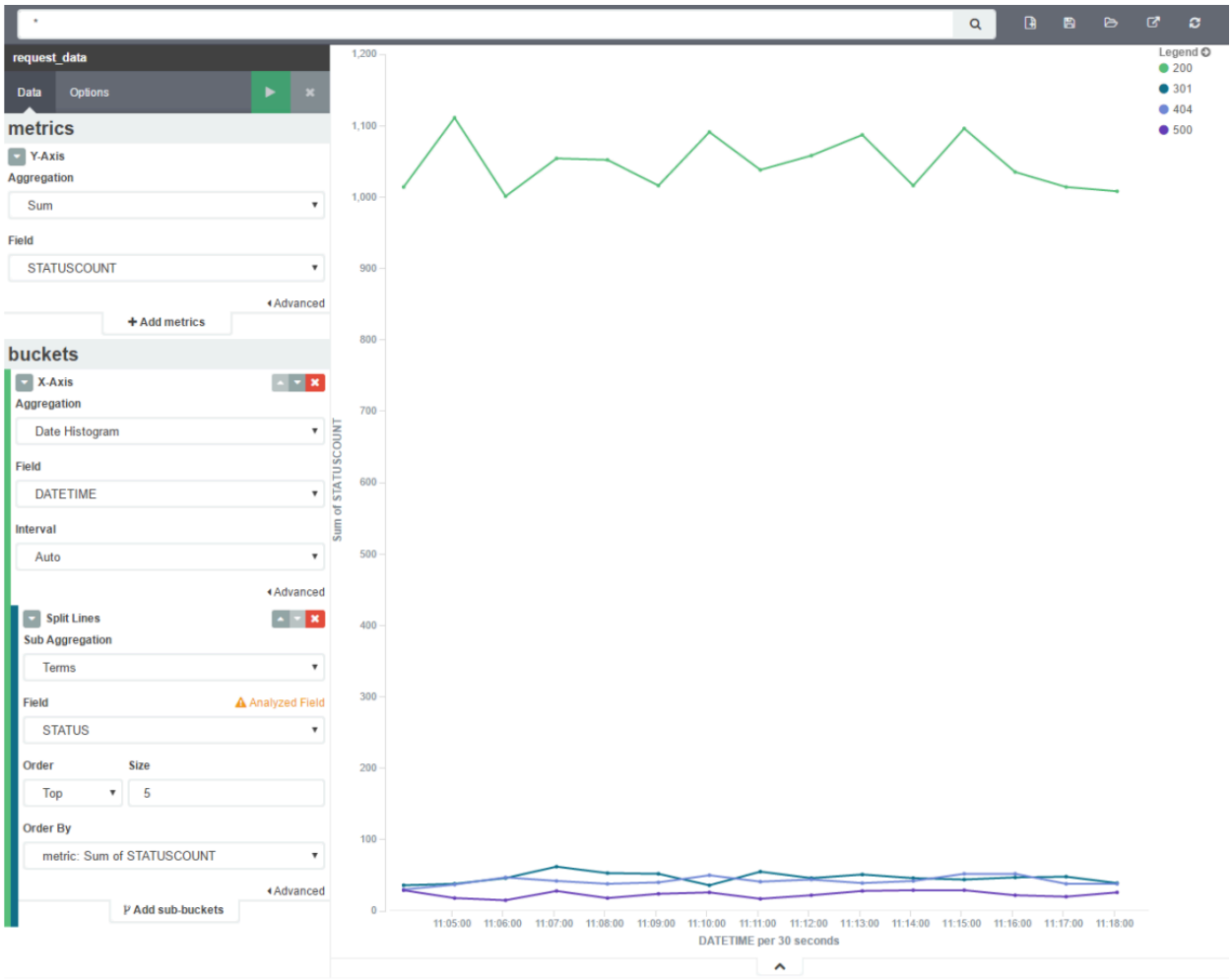
Step 7: Creating an Amazon Kinesis Analytics Application
Having created the second delivery stream and the corresponding endpoint (Amazon ES), I then created the Amazon Kinesis Analytics application to aggregate data from the streaming log data and store it into the Amazon ES domain. To aggregate the incoming data, I have configured a SQL Query in Amazon Kinesis Analytics. This query creates an aggregated dataset each minute, for further analysis.


Instances | EC2 Management
S3 Management Console
Amazon Kinesis Analytics
Amazon Kinesis Firehose
kinesis firehose empty incor
Monitoring the Amazon Kine
us-east-2.console.aws.amazon.com/kinesisanalytics/home?region=us-east-2#/applications/dashboard
aws
Services
Resource Groups
karansawsacc
Ohio
Support
Amazon Kinesis
Dashboard
Data Streams
Data Firehose
Data Analytics
Video Streams
External resources
What's new
Kinesis Data Analytics applications
Kinesis Data Analytics applications continuously read and process data from streaming sources in real-time. Learn more
Create application
Actions
Filter or search by application name
Viewing 1 - 1 of 1 applications
Application name
Runtime
State
web-log-aggregation-tutorial
SQL
Running
Input
Source ARN arn:aws:firehose:us-east-2:177843202009:deliverystream/web-log-ingestion-stream
Role ARN arn:aws:iam::177843202009:role/service-role/kinesis-analytics-web-log-aggregation-tutorial-us-east-2
Format JSON
Created May 27, 2020 10:17:19 PM
Last updated May 27, 2020 11:52:38 PM
Application details
Output
Destination ARN arn:aws:firehose:us-east-2:177843202009:deliverystream/web-log-aggregated-data
Role ARN arn:aws:iam::177843202009:role/service-role/kinesis-analytics-web-log-aggregation-tutorial-us-east-2
Format JSON
Viewing 1 - 1 of 1 applications
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


Instances | EC2 Management
S3 Management Console
Amazon Kinesis Analytics
Amazon Kinesis Firehose
kinesis firehose empty incor
Monitoring the Amazon Kine
us-east-2.console.aws.amazon.com/kinesisanalytics/home?region=us-east-2#/wizard/editor?applicationName=web-log-aggregation-tutorial
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
8 *
9 * Get started by clicking "Add SQL from templates" or pull up the
10 * documentation and start writing your own custom queries.
11 */
12 CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (datetime VARCHAR(30), status INTEGER, statusCount INTEGER);
13 CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
14 SELECT STREAM TIMESTAMP_TO_CHAR('yyyy-MM-dd''T''HH:mm:ss.SSS',LOCALTIMESTAMP) as datetime, "response" as status,
15 COUNT(*) AS statusCount
16 FROM "SOURCE_SQL_STREAM_001"
17 GROUP BY "response", FLOOR(("SOURCE_SQL_STREAM_001".ROWTIME - TIMESTAMP '1970-01-01 00:00:00') minute / 1 TO MINUTE);
18
Application status: RUNNING
Source
Real-time analytics
Destination
In-application streams:
DESTINATION_SQL_STREAM
error_stream
Pause results
New results are added every 2-10 seconds. The results below are sampled.
Scroll to bottom when new results arrive.
Filter by column name
ROWTIME
DATETIME
STATUS
STATUSCOUNT
2020-05-28 04:49:00.0 2020-05-28T04:48:59.989 200 914
2020-05-28 04:50:00.0 2020-05-28T04:48:59.989 200 2709
2020-05-28 04:50:00.0 2020-05-28T04:48:59.989 301 115
2020-05-28 04:50:00.0 2020-05-28T04:48:59.989 404 109
2020-05-28 04:50:00.0 2020-05-28T04:48:59.989 500 67
2020-05-28 04:51:00.0 2020-05-28T04:48:59.989 200 6785

Step 8: Viewing the Aggregated Streaming Data

After approximately 5 minutes, data (output from SQL Statement) gets written into Amazon ES. From here, ES has built-in support for Kibana, a tool that allows users to explore and visualize the data stored in an Elasticsearch cluster. Using the link to Kibana provided in Elasticsearch, I then created and configured a line chart in Kibana to show how many of each HTTP response type were included in the source web log data per minute.



After all the steps were completed, and the streaming logs were visible in Kibana in real-time, I terminated and teared down all the services to avoid incurring any costs.