

Amazon Athena Cookbook

June 2017



Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

| | |
|--|--------------------|
| Introduction..... | 5 |
| Cloudtrail Logs..... | 6 |
| Creating the Cloudtrail table..... | 6 |
| Querying Cloudtrail logs | 8 |
| Additional Cloudtrail reading | 8 |
| Cloudfront Logs | 9 |
| Creating the Cloudfront table | 9 |
| Querying Cloudfront logs..... | 10 |
| Additional Cloudfront reading..... | 10 |
| Elastic Load Balancer Classic Logs | 11 |
| Creating the ELB table | 11 |
| Querying ELB logs | 12 |
| Additional ELB reading | 12 |
| Application Load Balancer Logs..... | 13 |
| Creating the ALB table | 13 |
| Querying ALB logs | 14 |
| Additional ALB reading | 14 |
| VPC Flow Logs..... | 15 |
| Creating the VPC Flow table | 15 |
| Querying VPC flow logs..... | 16 |
| Additional VPC Flow reading..... | 16 |

Abstract

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run.

Introduction

Athena is easy to use. Simply point to your data in Amazon S3, define the schema, and start querying using standard SQL. Most results are delivered within seconds. With Athena, there's no need for complex ETL jobs to prepare your data for analysis. This makes it easy for anyone with SQL skills to quickly analyze large-scale datasets.

This cookbook will present several recipes for getting started by leveraging Athena to query popular datasets. We will use the Athena console but you can also use other tools that connect via JDBC, such as [SQL Workbench](#) or the AWS [CLI](#) and [APIs](#).

Before getting started, if you have not yet done so, please follow [these instructions](#) to setup an IAM user with appropriate permissions to access Athena and the S3 bucket where the data you want to query reside.

Cloudtrail Logs

Cloudtrail logs a great source of information regarding any API calls being made to your AWS services, including the console. Cloudtrail needs to be enabled for it to collect logs and save them to S3. Follow [these instructions](#) to get started enabling Cloudtrail. Note the destination S3 bucket where logs will be saved, it will be needed in the next section.

Creating the Cloudtrail table

Copy and paste the following DDL statement into the Athena console. Please make sure to modify the items highlighted in yellow.

```
CREATE EXTERNAL TABLE cloudtrail_logs (  
  eventversion STRING,  
  userIdentity STRUCT<  
    type:STRING,  
    principalid:STRING,  
    arn:STRING,  
    accountid:STRING,  
    invokedby:STRING,  
    accesskeyid:STRING,  
    userName:String,  
    sessioncontext:STRUCT<  
      attributes:STRUCT<  
        mfaauthenticated:STRING,  
        creationdate:STRING>,  
      sessionIssuer:STRUCT<  
        type:STRING,  
        principalId:STRING,  
        arn:STRING,  
        accountId:STRING,  
        userName:STRING>>>,  
  eventTime STRING,  
  eventSource STRING,  
  eventName STRING,  
  awsRegion STRING,  
  sourceIpAddress STRING,  
  userAgent STRING,  
  errorCode STRING,  
  errorMessage STRING,  
  requestId STRING,  
  eventId STRING,  
  resources ARRAY<STRUCT<  
    ARN:STRING,  
    accountId:STRING,  
    type:STRING>>,  
  )
```

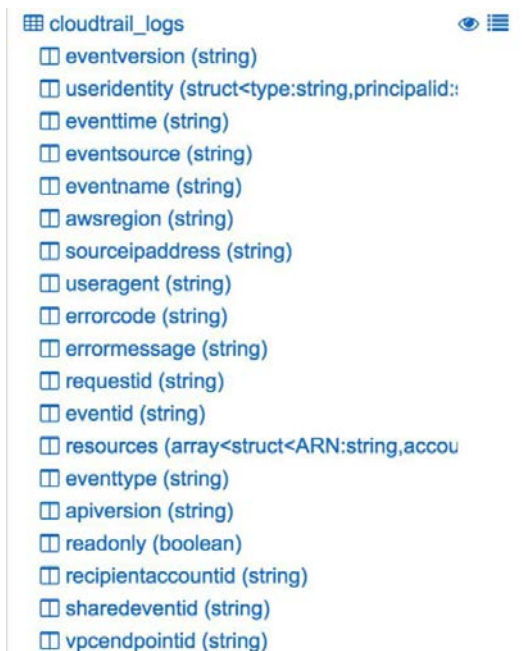
```
eventType STRING,  
apiVersion STRING,  
readOnly BOOLEAN,  
recipientAccountId STRING,  
sharedEventID STRING,  
vpcEndpointId STRING  
)  
ROW FORMAT SERDE 'com.amazon.emr.hive.serde.CloudTrailSerde'  
STORED AS INPUTFORMAT  
'com.amazon.emr.cloudtrail.CloudTrailInputFormat'  
OUTPUTFORMAT  
'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'  
LOCATION 's3://cloudtrail_bucket_name/AWSLogs/112233445566/'
```

Once you've finished making the changes, press the



Once the query completes and the table definition has been successfully registered with Athena, the data is now ready to be queried.

On the left hand side of the Athena console you will see a list of tables in the “default” database. Click the grid icon to the left of the cloudtrail_logs table to expand the list of columns. It should look like the following:



| cloudtrail_logs |
|---|
| <input type="checkbox"/> eventversion (string) |
| <input type="checkbox"/> useridentity (struct<type:string,principalid:: |
| <input type="checkbox"/> eventtime (string) |
| <input type="checkbox"/> eventsource (string) |
| <input type="checkbox"/> eventname (string) |
| <input type="checkbox"/> awsregion (string) |
| <input type="checkbox"/> sourceipaddress (string) |
| <input type="checkbox"/> useragent (string) |
| <input type="checkbox"/> errorcode (string) |
| <input type="checkbox"/> errormessage (string) |
| <input type="checkbox"/> requestid (string) |
| <input type="checkbox"/> eventid (string) |
| <input type="checkbox"/> resources (array<struct<ARN:string,accou |
| <input type="checkbox"/> eventtype (string) |
| <input type="checkbox"/> apiversion (string) |
| <input type="checkbox"/> readonly (boolean) |
| <input type="checkbox"/> recipientaccountid (string) |
| <input type="checkbox"/> sharedeventid (string) |
| <input type="checkbox"/> vpcendpointid (string) |

Querying Cloudtrail logs

Verify that your table looks the same as the one shown here. If it isn't first delete the existing table as so: ***DROP TABLE cloudtrail_logs;*** Then recreate the table as shown in the previous section.

To explore the Cloudtrail data start by looking at which IAM users called which APIs and from which source IP addresses.

Copy and paste the following SQL query to the Athena console and click Run Query

```
SELECT useridentity.arn, eventname, sourceipaddress, eventtime
FROM cloudtrail_logs
LIMIT 100;
```

That's it for this recipe; feel free to explore your data further. Remember to include a LIMIT clause to tell Athena to only return a subset of the rows. This is in case a query goes awry.

Additional Cloudtrail reading

If you are interested in learning more about querying Cloudtrail logs with Athena take a look at this [blog post](#).

Cloudfront Logs

Cloudfront CDN can be configured to export web distribution access logs to S3. These logs allow you to explore users' surfing patterns across your web properties served by Cloudfront. Follow [these instructions](#) to enable Cloudfront web distribution access log on your preferred Cloudfront distribution. Please note this recipe will not work for RTMP distribution logs. Make a note of the S3 bucket where these logs will be saved.

Creating the Cloudfront table

Copy and paste the following DDL statement into the Athena console. Please make sure to modify the items highlighted in yellow.

```
CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront_logs (  
  `date` date,  
  `time` string,  
  `location` string,  
  bytes bigint,  
  requestip string,  
  method string,  
  host string,  
  uri string,  
  status int,  
  referrer string,  
  useragent string,  
  querystring string,  
  cookie string,  
  resulttype string,  
  requestid string,  
  hostheader string,  
  requestprotocol int,  
  requestbytes bigint,  
  timetaken bigint,  
  xforwardedfor string,  
  sslprotocol string,  
  sslcipher string,  
  resporesulttype string,  
  httpversion string
```


Elastic Load Balancer Classic Logs

Elastic Load Balancer logs are a convenient place to analyze and understand traffic patterns to and from ELBs and backend applications. You can see the source of traffic, latency and bytes transferred. First follow [these instructions](#) to enable ELB logs that will be saved to a destination S3 bucket.

Creating the ELB table

Copy and paste the following DDL statement into the Athena console. Please make sure to modify the items highlighted in yellow.

```
CREATE EXTERNAL TABLE IF NOT EXISTS elb_logs (
  request_timestamp string,
  elb_name string,
  request_ip string,
  request_port int,
  backend_ip string,
  backend_port int,
  request_processing_time double,
  backend_processing_time double,
  client_response_time double,
  elb_response_code string,
  backend_response_code string,
  received_bytes bigint,
  sent_bytes bigint,
  request_verb string,
  url string,
  protocol string,
  user_agent string,
  ssl_cipher string,
  ssl_protocol string )
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  'serialization.format' = '1', 'input.regex' = '([^\ ]*) ([^\ ]*) ([^\ ]*) : ([0-9]*) ([^\ ]*) [:-] ([0-9]*) ([-0-9]*) ([-0-9]*) ([-0-9]*) ([-0-9]*) (-|[-0-9]*) ([-0-9]*) ([-0-9]*) \\\\"([^\ ]*) ([^\ ]*) (-|[-0-9]*) \\\\" ([^\ ]*) ([^\ ]*) ([A-Z0-9-]*) ([A-Za-z0-9-]*)$' )
LOCATION
's3://your_log_bucket/prefix/AWSLogs/112233445566/elasticloadbalancing/';
```

Once the query completes and the table definition has been successfully registered with Athena, the data is now ready to be queried.

Querying ELB logs

In the following query you will list the backend application servers that returned a 4XX or 5XX error response code.

```
SELECT request_timestamp, elb_name, backend_ip, backend_response_code
FROM elb_logs
WHERE backend_response_code LIKE '4%' OR backend_response_code LIKE
'5%'
LIMIT 100;
```

The next query will sum up the response time of all the transactions grouped by backend IP address and ELB name.

```
SELECT sum(backend_processing_time) as total_ms, elb_name, backend_ip
FROM elb_logs
WHERE backend_ip <> ''
GROUP BY backend_ip, elb_name
LIMIT 100;
```

Additional ELB reading

If you are interested in learning more about querying ELB logs with Athena take a look at this [blog post](#).

Application Load Balancer Logs

Application Load Balancer is a new type of load balancer that enables traffic distribution in a microservices deployment that utilizes containers. This recipe will inspect ALB logs, similar to ELB logs in the previous recipe. Follow [these instructions](#) to enable ALB access logs.

Creating the ALB table

Copy and paste the following DDL statement into the Athena console. Please make sure to modify the items highlighted in yellow.

```
CREATE EXTERNAL TABLE IF NOT EXISTS alb_logs (  
  type string,  
  time string,  
  elb string,  
  client_ip string,  
  client_port string,  
  target string,  
  request_processing_time int,  
  target_processing_time int,  
  response_processing_time int,  
  elb_status_code int,  
  target_status_code string,  
  received_bytes int,  
  sent_bytes int,  
  request_verb string,  
  request_url string,  
  request_proto string,  
  user_agent string,  
  ssl_cipher string,  
  ssl_protocol string,  
  target_group_arn string,  
  trace_id string  
)
```

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  'serialization.format' = '1',
  'input.regex' = '([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^ ]*) ([-0-9]*) ([-0-9]*) ([-0-9]*) ([-0-9]*) ([^ ]*) ([-0-9]*) ([-0-9]*) \"([^ ]*) ([^ ]*) ([^ ]*)\" \"([^\"]*)\" ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)'
) LOCATION
's3://your_log_bucket/prefix/AWSLogs/112233445566/elasticloadbalancing/'
;
```

Once the query completes and the table definition has been successfully registered with Athena, the data is now ready to be queried.

Querying ALB logs

The following query will count how many HTTP GET requests were received by the load balancer grouped by client IP address.

```
SELECT COUNT(request_verb) AS count, request_verb, client_ip
FROM alb_logs
GROUP BY request_verb, client_ip
LIMIT 100;
```

Another interesting query would be to see which URLs do Safari browser users visit.

```
SELECT request_url
FROM alb_logs
WHERE user_agent LIKE '%Safari%'
LIMIT 10;
```

Additional ALB reading

If you are interested in learning more about querying ALB logs with Athena take a look at this [blog post](#).

Querying VPC Flow logs

The following query will list all of the rejected TCP connections. You can see in this query we're using a presto [datetime function](#) to convert the timestamp field ("ts") and extract only the day of the week for which these events occurred.

```
SELECT day_of_week(from_iso8601_timestamp(ts)) as day, interfaceid,
sourceaddress, action, protocol
FROM vpc_flow_logs
WHERE action = 'REJECT' AND protocol = 6
LIMIT 100;
```

Next, count the number of packets received on HTTPS port 443 and group them by destination IP address and return the top 10. We can then see which one of our servers is receiving the most HTTPS traffic.

```
SELECT SUM(numpackets) AS packetcount, destinationaddress FROM
vpc_flow_logs
WHERE destinationport = 443
GROUP BY destinationaddress
ORDER BY packetcount DESC
LIMIT 10;
```

Additional VPC Flow reading

If you are interested in learning more about querying VPC flow logs with Athena take a look at this [blog post](#).