Audit logs migration guide

Overview

BigQuery currently provides two versions of audit logs: an older version that uses `AuditData` payload, and a new version that uses `BigQueryAuditMetadata`. This document describes migrating logs interface filters, BigQuery exports, and changes in queries over the exported logs from the old format to the new format.

Logs interface filters

Changing filters might require changing the paths to the fields you want to filter on. BigQuery audit logs overview (/bigquery/docs/reference/auditlogs) lists the changes between the old logs and the new logs. The changes include the `resource`, `resourceName`, and `methodName` fields.

For example, the filter to select all the BigQuery-related records changes from:

```
rce.type = "bigquery_resource"
```

to:

```
rce.type = ("bigquery_project" OR "bigquery_dataset")
```

For the filter table inserts on the method name, the filter changes from:

```
Payload.methodName="tableservice.insert"
```

to:

```
Payload.methodName = "google.cloud.bigquery.v2.TableService.InsertTable"
```

You can also easily select all table inserts regardless of the method:

```
Payload.metadata.tableCreation.reason != ""
```

Or filter on a specific reason for a table creation:

```
Payload.metadata.tableCreation.reason = "JOB"
```

New logs have separate method names for patch and update, so

```
Payload.methodName = "tableservice.update"
```

changes to:

```
Payload.methodName = ("google.cloud.bigquery.v2.TableService.UpdateTable" OR "google
```

However, you can find all the table updates by using the **`protoPayload.metadata.tableChange.reason`** field.

Finally, with the new logs you can find all the records for a specific table by using the **`protoPayload.methodName`** field. For example:

```
Payload.resourceName = "projects/myproject/datasets/mydataset/tables/mytable"
```

If you're using a **`resourceName`** filter with the old logs to find all the side effects for a specific job ID, with the new logs you'll need to filter on a specific event instead. To find all the source tables read records for a specific job:

```
Payload.metadata.tableDataRead.jobName = "projects/myproject/jobs/myjob"
```

Or to find the destination table change:

```
Payload.metadata.tableChange.jobName = "projects/myproject/jobs/myjob"
```

Resources representations, such as Job or Table, in the new logs have the structure similar to the old logs. Hovewer, new logs represent events while the old logs focus more on the request and response.

You can compare the new BigQueryAuditMetadata (/bigquery/docs/reference/auditlogs/rest/Shared.Types/BigQueryAuditMetadata) and the old AuditData (/bigquery/docs/reference/auditlogs/rest/Shared.Types/AuditData) formats. BigQuery-specific information moved from the `serviceData` to the `metadata` field.

For example, to migrate a filter that finds all completed "CREATE TABLE AS SELECT" DDL jobs, change the filter from:

```
Payload.serviceData.jobCompletedEvent.job.jobConfiguration.query.statementType = "CR
```

to:

```
Payload.metadata.jobChange.after = "DONE"
Payload.metadata.jobChange.job.jobConfig.queryConfig.statementType = "CREATE_TABLE_A
```

Logs routing (exports)

For exporting all records for core BigQuery operations, use the metadata type filter:

```
Payload.metadata."@type"="type.googleapis.com/google.cloud.audit.BigQueryAuditMetada
```

Querying logs exported to BigQuery

In addition to the basic structure changes described earlier, there are additional changes in the exported data structure that need to be addressed. The **`metadata`** field is exported as a single JSON column. To query the contents, you must use BigQuery JSON functions (/bigquery/docs/reference/standard-sql/json_functions).

Example: DDL queries

For the previous example of filtering all the "CREATE TABLE AS SELECT" DDL jobs, the query over the old logs to count the number of jobs might look like this:

```
andardSQL
ECT COUNT(*)
M
MYPROJECTID.MYDATASETID.cloudaudit_googleapis_com_data_access_YYYYMMDD`
RE
rotopayload_auditlog.servicedata_v1_bigquery.jobCompletedEvent.job.jobConfiguration.
```

The same query over the new logs might look like this:

```
ECT
OUNT(*)
M
MYPROJECTID.MYDATASETID.cloudaudit_googleapis_com_data_access_YYYYMMDD`
RE
SON_EXTRACT_SCALAR(protopayload_auditlog.metadataJson,
 "$.jobChange.job.jobConfig.queryConfig.statementType") = "CREATE_TABLE_AS_SELECT"
```

Example: Hourly cost breakdown

Old query:

```
andardSQL
ECT
IMESTAMP_TRUNC(protopayload_auditlog.servicedata_v1_bigquery.jobCompletedEvent.job.j
ORMAT('%9.2f',5.0 * (SUM(protopayload_auditlog.servicedata_v1_bigquery.jobCompletedE
M
MYPROJECTID.MYDATASETID.cloudaudit_googleapis_com_data_access_YYYYMMDD`
```

```
RE
rotopayload_auditlog.servicedata_v1_bigquery.jobCompletedEvent.eventName = 'query_jo
UP BY time_window
 ORDER BY time_window DESC
```

New query:

```
ECT
IMESTAMP_TRUNC(TIMESTAMP(JSON_EXTRACT_SCALAR(protopayload_auditlog.metadataJson,
     "$.jobChange.job.jobStats.endTime")), HOUR) AS time_window,
ORMAT('%9.2f',5.0 * (SUM(CAST(JSON_EXTRACT_SCALAR(protopayload_auditlog.metadataJson
         "$.jobChange.job.jobStats.queryStats.totalBilledBytes") AS INT64))/POWER(2,
M
MYPROJECTID.MYDATASETID.cloudaudit_googleapis_com_data_access_YYYYMMDD`
RE
SON_EXTRACT_SCALAR(protopayload_auditlog.metadataJson,
 "$.jobChange.job.jobConfig.type") = "QUERY"
UP BY
ime_window
ER BY
ime_window DESC
```

Last updated 2020-06-26 UTC.