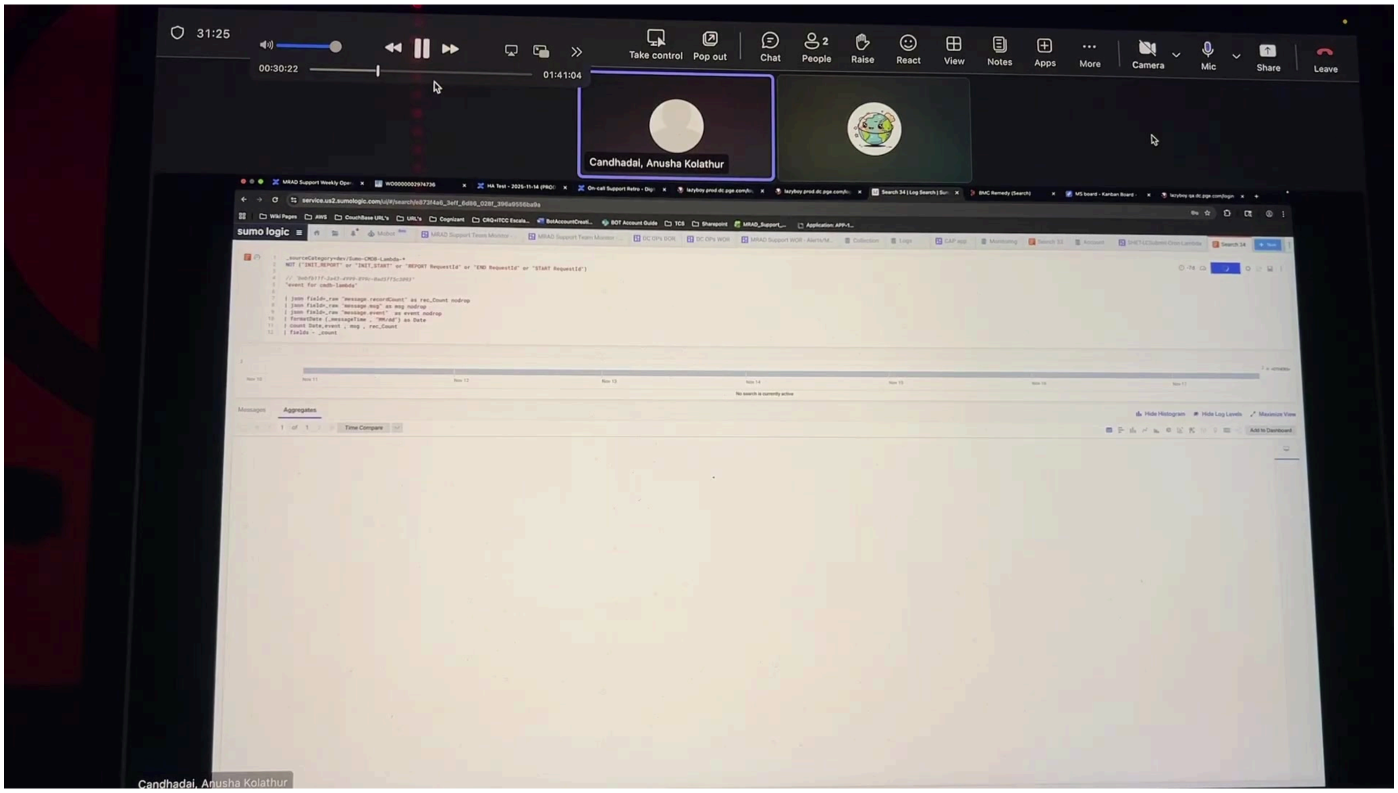
dash2 (cmdb lambda use case).mp4_75 | Summarize Videos, Audio, PDF & Websites

lilys.ai/digest/6961109/7339007



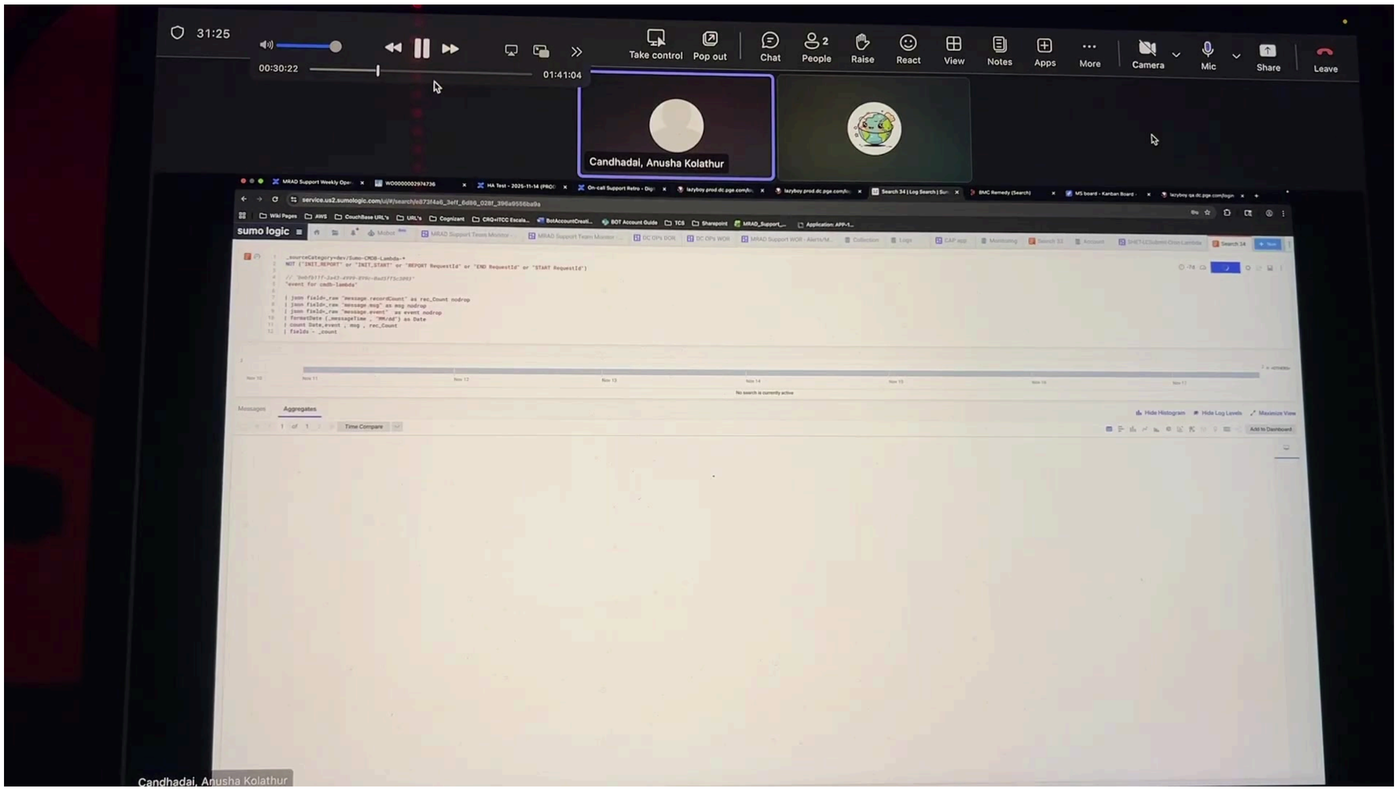
```
_sourceCategory=dev/Sumo-CMDB-Lambda-*
NOT ("INIT_REPORT" or "INIT_START" or "REPORT RequestId" or "END RequestId" or "START RequestId")

// "bebfb11f-3a43-4999-899c-8ad5ff5c3093"
"event for cmdb-lambda"

| json field=_raw "message.recordCount" as rec_Count nodrop
| json field=_raw "message.msg" as msg nodrop
| json field=_raw "message.event"  as event nodrop
| formatDate (_messageTime , "MM/dd") as Date
| count Date,event , msg , rec_Count
| fields - _count
```

Table of Contents

Master the art of **CMDB Lambda monitoring** by learning to build precise, actionable dashboards from raw logs. This guide shows you exactly how to query logs to create a **daily success summary** and a detailed panel tracking **input, unique, and duplicate record counts**. Stop guessing about performance; learn the specific querying techniques to transform confusing logs into clear operational insights for better troubleshooting.

1. Initial Dashboard Setup and Success Summary Panel Creation [1]

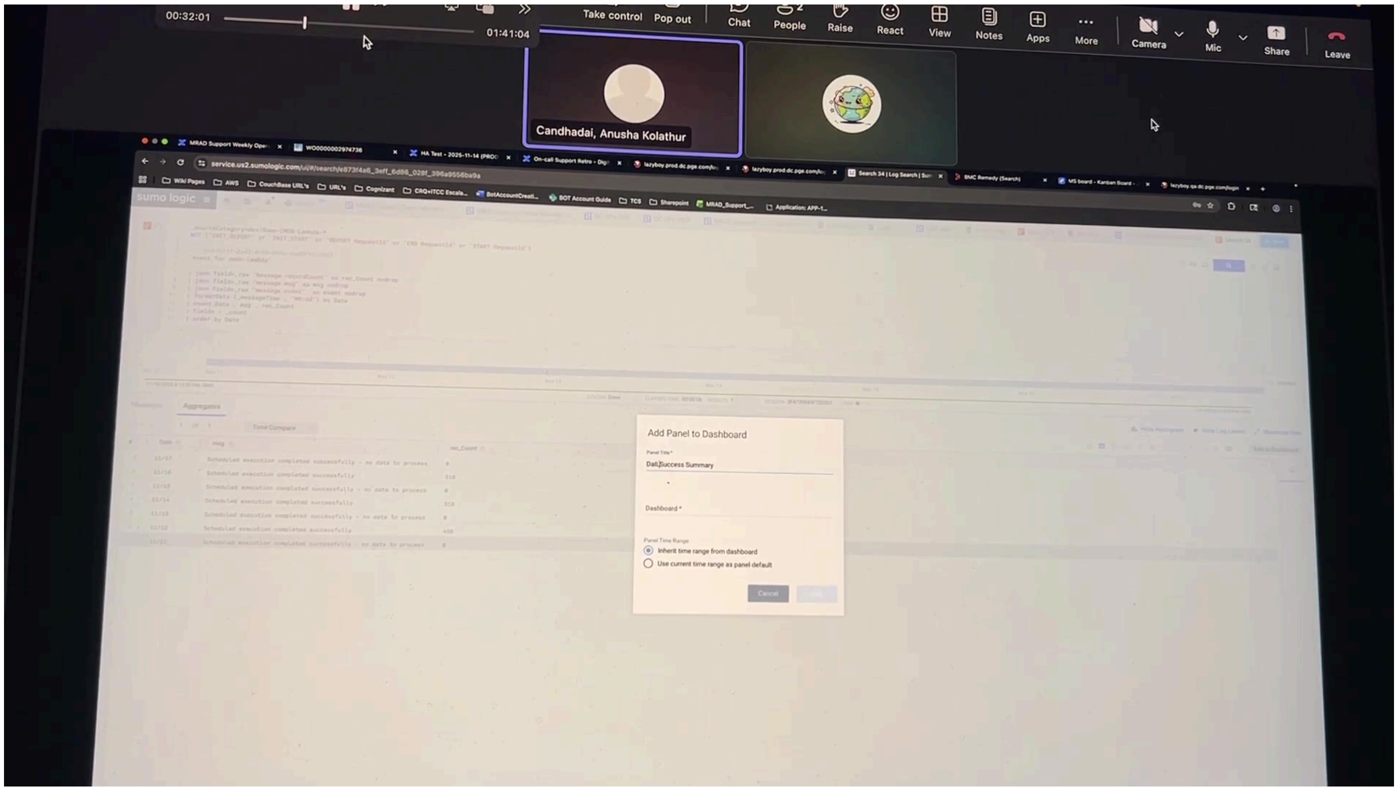
00:00:01 (3 min)



```
_sourceCategory=dev/Sumo-CMDB-Lambda-*
NOT ("INIT_REPORT" or "INIT_START" or "REPORT RequestId" or "END RequestId" or "START RequestId")

// "bebfb11f-3a43-4999-899c-8ad5ff5c3093"
"event for cmdb-lambda"

| json field=_raw "message.recordCount" as rec_Count nodrop
| json field=_raw "message.msg" as msg nodrop
| json field=_raw "message.event"  as event nodrop
| formatDate (_messageTime , "MM/dd") as Date
| count Date,event , msg , rec_Count
| fields - _count
```

1. **Field Removal**: The presenter removes the `_count` field from the query output. [1]
2. **Panel Preparation**: This modified query output is used to demonstrate the dashboard panel look. [4]
3. **Formatting**: The panel is neatened by ordering results by date. [6]
4. **Source Context**: The current dashboard is based on **dev logs** for now. [7]
5. **Future Parameterization**: The presenter plans to show how to add parameters later. [8]
6. **Successful Run Identification**: The first panel identifies if the run was successful on a given date. [13]
7. **Useless Events**: An event showing successful completion with nothing to process is deemed useless. [14]
8. **Desired Success Message**: The desired message for success is: "schedule execution completed successfully, schedule, schedule, schedule". [19]

9. **Adding to Dashboard**: After running the query, the user clicks "add to dashboard". [21]
10. **Panel Naming**: The panel is named **Success summary** because it is sorted daily. [22]



```
_sourceCategory=dev/Supp-CMDB-Lambda-*
NOT ("INIT_REPORT" or "INIT_START" or "REPORT RequestId" or "END RequestId" or "START RequestId")
"event for cmdb-lambda"
| json field=_raw "message.recordCount" as rec_Count nodrop
| json field=_raw "message.msg" as msg nodrop
| json field=_raw "message.event" as event nodrop
| formatDate (_messageTime , "MM/dd") as Date
| count Date , msg , rec_Count
| fields - _count
| order by Date
```

Add Panel to Dashboard

Panel Title*: DailSuccess Summary

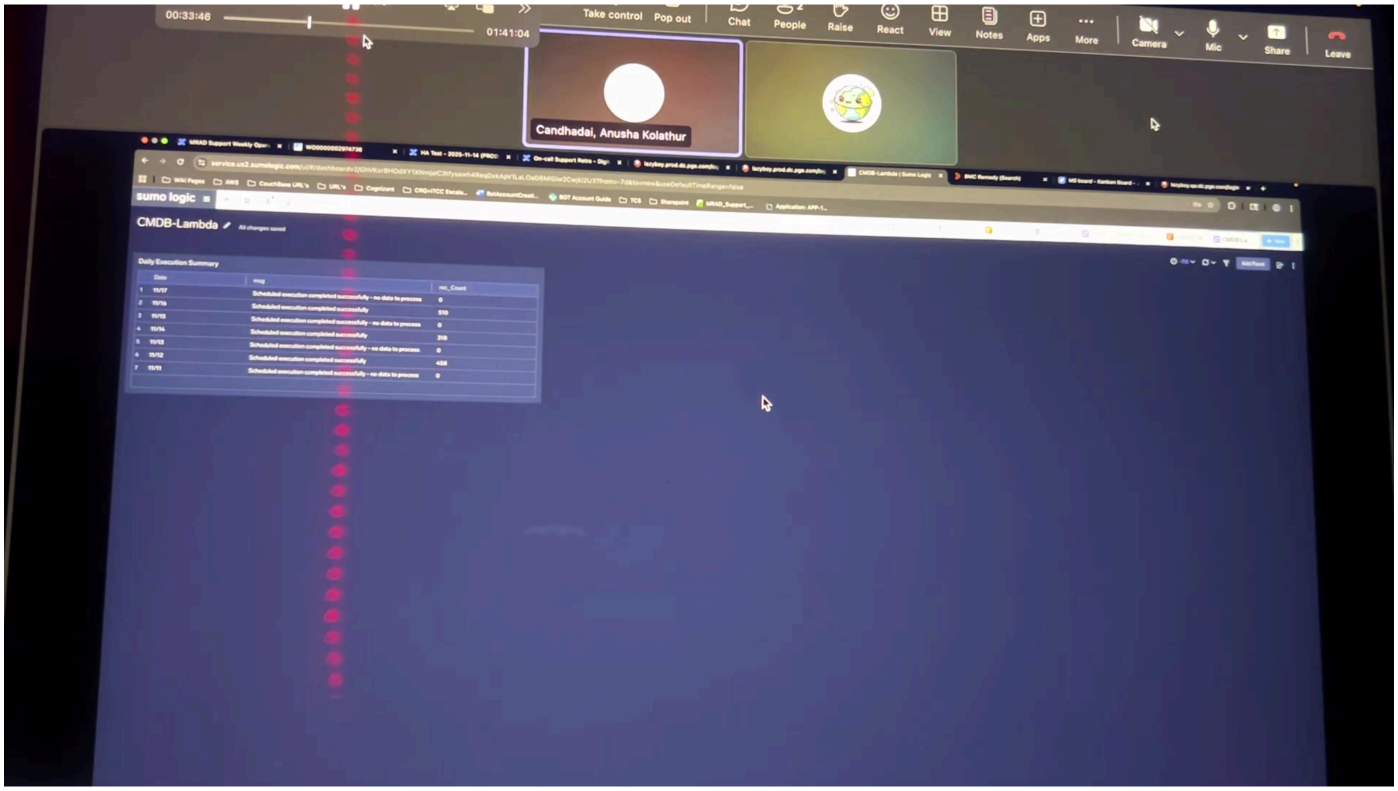
Dashboard *

Panel Time Range
[✓ Inherit time range from dashboard]
Use current time range as panel default

1. **Dashboard Creation**: A new dashboard is created for the **CMDB lambda**. [25]
2. **Location**: The panel is added under the `support to emerald AWS` section. [28]
3. **Default View**: The new dashboard defaults to showing the last **15 minutes**. [32]
4. **Time Range Adjustment**: The time range is adjusted to **seven days**. [34]
5. **Panel Validation**: The first panel shows daily execution status. [35]
6. **Valid Non-Processing Run**: A run on the 11th showed it ran but had nothing to process, which is valid. [37]
7. **Panel Naming Refinement**: The panel name is refined from "daily success summary" to **daily summary**. [39]
8. **Purpose**: This panel gives an idea if the Lambda execution was successful or not. [44]

2. Creating the Counts Panel for Input, Unique, and Duplicate Records [45]

00:03:25 (4 min)



Daily Execution Summary

	Date	msg	rec_Count
1	11/17	Scheduled execution completed successfully - no data to process	0
2	11/16	Scheduled execution completed successfully	510
3	11/15	Scheduled execution completed successfully - no data to process	0
4	11/14	Scheduled execution completed successfully	318
5	11/13	Scheduled execution completed successfully - no data to process	0
6	11/12	Scheduled execution completed successfully	458
7	11/11	Scheduled execution completed successfully - no data to process	0

1. **Next Panel Goal**: The next important panel must show the **counts** of processed records. [45]
2. **Goal Detail**: Determine how many of the total records were processed versus how many were duplicates. [45]
3. **Query Reset**: The presenter goes back to the log and removes all previous filters to start querying again. [46]
4. **Querying Best Practice**: Always query using a **single transaction or document ID** first for easier query framing. [47]
5. **Pre-Panel Step**: Before adding to the panel, the specific transaction ID filter must be removed. [52]
6. **Input Record Count Source**: The **input record count** is found under the "query executed successfully" log. [52]
7. **Observed Input Count**: For one example, the input record count was **510**. [53]

8. **Filtering Unnecessary Events**: Events like "database query completed" are filtered out as they are FYI information. [56]

9. **Focus**: The focus is only on the count log to name it the **input record count**. [61]

Extracting capture image...

1. **Duplicate Handling Log**: The duplicate handling mechanism is found in the event: `duplicates remote`. [64]

2. **Duplicate Data**: This log states: 2065 unique values and 304 were duplicate out of 500 original items. [65]

3. **Parsing Goal**: The goal is to parse out the **input, unique, and duplicate** counts into one panel. [66]

4. **Log Verification**: The presenter checks other logs to ensure no other relevant information exists. [67]

5. **Duplicate Detail**: Another log shows 206 good ones, 2 new, and 204 existing records. [70]

6. **Troubleshooting Scope**: Showing the exact breakdown of new vs. existing documents is for *troubleshooting*, not the main panel. [71]

7. **Final Log Check**: No other logs provide meaningful information for this count panel. [78]

8. **Interesting Log**: An interesting log found was: "load uploaded, updated free with a total." [81]

9. **Next Step**: The presenter will parse the **unique count** and the **duplicate** count logs. [83]

10. **Naming**: The unique count log will be named **unique record count**. [85]

11. **Conclusion**: This specific log set provides the input, unique, and duplicate counts. [87]

3. Aggregating Counts and Finalizing the Counts Panel [89]

00:07:58 (6 min)

Extracting capture image...

1. **Further Detail Consideration**: A log shows how many unique records were *updated* versus *uploaded*. [89]

2. **Complexity Decision**: The presenter questions if showing update/upload status will confuse the view. [90]

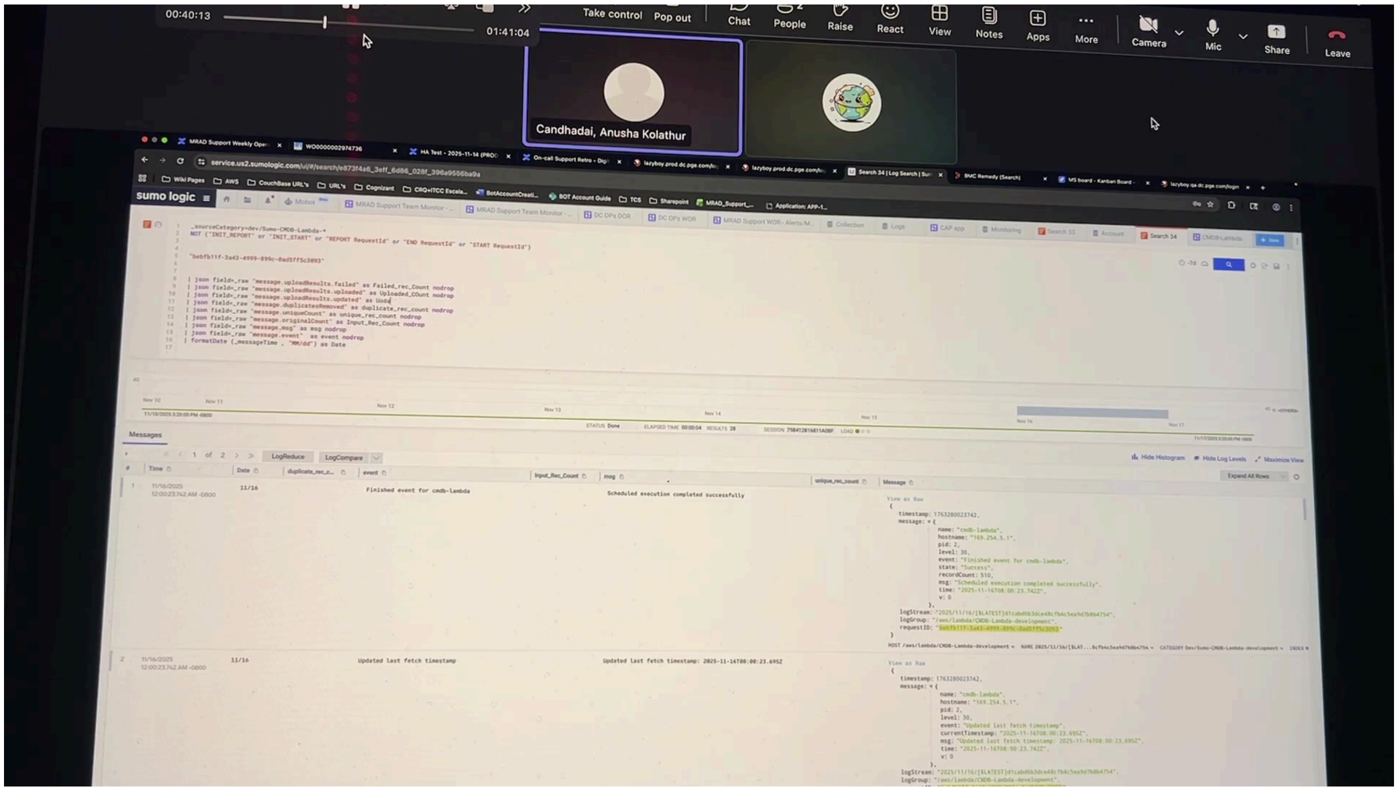
3. **Table Format Decision**: They decide to parse it out and put it in a table format first. [91]

4. **Panel Execution Timing**: Panel queries in Sumo Logic are **real-time** by default. [95]

5. **Real-Time Behavior**: When the dashboard is opened, the query executes, showing data up to that moment. [97]

6. **Scheduling Note**: It is *not* possible to schedule the panel execution itself. [102]

7. **New Counts to Parse**: The presenter looks for **uploader count** and **updated** counts. [107]



```
_sourceCategory=dev/Sumo-CMDB-Lambda-*
NOT ("INIT_REPORT" or "INIT_START" or "REPORT RequestId" or "END RequestId" or "START RequestId")


"bebfb11f-3a43-4999-899c-0ad5ff5c3093"


| json field=_raw "message.uploadResults.failed" as Failed_rec_Count nodrop
| json field=_raw "message.uploadResults.uploaded" as Uploaded_COunt nodrop
| json field=_raw "message.uploadResults.updated" as Uoda
| json field=_raw "message.duplicatesRemoved" as duplicate_rec_count nodrop
| json field=_raw "message.uniqueCount" as unique_rec_count nodrop
| json field=_raw "message.originalCount" as Input_Rec_Count nodrop
| json field=_raw "message.msg" as msg nodrop
| json field=_raw "message.event"  as event nodrop
| formatDate (_messageTime , "MM/dd") as Date
```

```
1  11/16/2025 12:00:23.742 AM -0800  11/16  Finished event for cmdb-lambda  Scheduled execution completed successfully
View as Raw
{
   timestamp: 1763280023742,
   message: ▾{
         name: "cmdb-lambda",
         hostname: "169.254.5.1",
         pid: 2,
         level: 30,
         event: "Finished event for cmdb-lambda",
         state: "Success",
         recordCount: 510,
         msg: "Scheduled execution completed successfully",
         time: "2025-11-16T08:00:23.742Z",
         v: 0
      },
   logStream: "2025/11/16/[$LATEST]d1cabd6b3dce48cfb4c5ea9d7b8b4754",
   logGroup: "/aws/lambda/CMDB-Lambda-development",
   requestID: "bebfb11f-3a43-4999-899c-0ad5ff5c3093"
}
HOST /aws/lambda/CMDB-Lambda-development ▾  NAME 2025/11/16/[$LAT...8cfb4c5ea9d7b8b4754 ▾  CATEGORY Dev/Sumo-CMDB-Lambda-development ▾

2  11/16/2025 12:00:23.742 AM -0800  11/16  Updated last fetch timestamp  Updated last fetch timestamp: 2025-11-16T08:00:23.695Z
View as Raw
{
   timestamp: 1763280023742,
   message: ▾{
         name: "cmdb-lambda",
         hostname: "169.254.5.1",
         pid: 2,
         level: 30,
         event: "Updated last fetch timestamp",
         currentTimestamp: "2025-11-16T08:00:23.695Z",
         msg: "Updated last fetch timestamp: 2025-11-16T08:00:23.695Z",
         time: "2025-11-16T08:00:23.742Z",
         v: 0
      },
   logStream: "2025/11/16/[$LATEST]d1cabd6b3dce48cfb4c5ea9d7b8b4754",
   logGroup: "/aws/lambda/CMDB-Lambda-development",
```

1. **Initial Visualization Issue**: Initial attempts to display all counts result in records split into rows. [127]

2. **Preferred Visualization**: A **table** format is required, not a line chart over time. [128]

3. **Aggregation Method**: To get results in a single row, they must **sum** the values. [132]

4. **Fields to Sum**: Summing includes: input record count, unique count, and duplicate record count. [134]

5. **Renaming Aggregated Fields**: Fields are renamed using aliases like `As total records` and `unique, copy`. [137]

6. **Data Source Discrepancy**: The failed record count comes from a different log hierarchy than the other three counts. [146]

7. **Aggregation Result**: Summing aggregates the results onto a single row showing totals for that day. [147]

8. **Expanding Search**: The search is expanded beyond the single transaction ID to seven days. [149]

9. **Example Aggregation**: For one transaction: 318 total, 232 unique, 86 duplicates. [152]

10. **Adding to Dashboard**: The finalized counts panel is added to the dashboard, ordered by date. [165]

4. Planning Subsequent Panels: Failure and All Events [166]

00:14:20 (4 min)

Extracting capture image...

1. **Panel Goal Confirmation**: The first panel created is the **daily execution summary**. [168]

2. **Panel Requirement**: Every Lambda needs a specific set of panels agreed upon internally. [172]

3. **Success Cases Covered**: So far, only success cases have been monitored. [184]

4. **Failure Panel Requirement**: The next panel must show **all failures or error messages**. [199]

5. **Current Log State**: There are currently *no failure scenarios* or proof of failure logs in Dev/QA. [188]

6. **Daily Execution Panel Scope**: The daily summary panel was *not* filtered for success or failure states. [194]

7. **Correction Plan**: Panels will be corrected as failure logs appear during ongoing monitoring. [200]

Extracting capture image...

1. **Next Panel**: The final panel planned is the **all events panel**. [202]
2. **Event States**: States include: *attempt* (beginning), *success*, and *skipped* (business decision). [214]
3. **All Events Content**: This panel shows the **event, message, and states** that occurred. [218]
4. **All Errors Panel**: A separate panel for **all errors** is standard practice, using a keyword match query. [227]
5. **Current Error State**: Currently, the **all errors** panel shows nothing. [223]
6. **Ignoring Unnecessary Logs**: To frame the all events panel, several logs must be ignored using a `nod` query operator. [231]
7. **Focusing the Query**: The focus returns to a **single transaction** to see available options (events and message). [239]
8. **Counting Issue**: Counting by event and message is problematic because the message content changes daily. [243]
9. **Message Variability Example**: Messages like "starting execution for 100 records" vary, making a simple count visually poor. [247]

5. Structuring the All Events Panel for Readability [251]

00:20:25 (7 min)

Extracting capture image...

1. **View Decision**: The presenter decides to focus on the **event** field only for now. [251]
2. **Scope Options**: Two options exist: date-based events or request ID-based events. [257]
3. **Request ID Value**: Request IDs are valuable because they are different for every transaction. [260]
4. **Date Preference**: The presenter prefers a **date thing** because users usually do not track request IDs. [258]
5. **Goal**: The goal is to show *what events came in on a specific day*. [271]
6. **Visualization Choice**: The **transpose row** view is chosen because it is *neater*. [274]

7. **Time Range for Review**: The view is set for the **last three days** for review purposes. [284]

Extracting capture image...

1. **Next Step**: The next step is **ordering the events** because they appear out of sequence (e.g., Initializing at number 11). [285]
2. **Ordering Method**: Ordering requires writing specific queries in Sumo Logic to rename events sequentially (1, 2, 3...). [287]
3. **Renaming Logic**: If the event matches a specific name (e.g., "initializing"), rename it to a number (e.g., "1"). [293]
4. **Benefit of Renaming**: Renaming makes the events legible and ensures correct ordering when sorting ascendingly. [300]
5. **Handling Ambiguous Names**: If an event name is used in multiple places (like "retrieving parameter"), the **message field** must also be checked for differentiation. [303]
6. **Counter Idea**: A suggestion is made to use a counter that adds +1 like an if/else loop. [306]
7. **Attempting Time Ordering**: An attempt to order by `message time` fails because the stored time format is complex. [313]
8. **Conclusion on Ordering**: The ordering process will unfortunately have to be **manual** by renaming events. [336]
9. **Final Status**: The presenter acknowledges the manual effort required to rename events like "retrieved parameter." [341]