dash3 (cmdb lambda use case).mp4_05 | Summarize Videos, Audio, PDF & Websites

Table of Contents

This session dives deep into **dash3 CMDB lambda use case** logic, revealing how to manage complex, ordered event processing in real-time systems. You will learn practical techniques for **forcing event ordering** using manual numbering and keyword matching within scheduled Lambda functions. Discover how to structure dashboards by filtering and grouping numerous events, even when facing unexpected failures or needing to refine parameter visibility.

1. CMDB Lambda Use Case Logic and Event Ordering [1]

00:00:06 (16 min)

Extracting capture image...

1. **Lambda Execution Context** [1]
    1. Most processing happens in *real-time* [1]
    2. This specific Lambda runs *once a day* [2]
    3. It is a **scheduled Lambda**, not a real-time one [4]

2. **Parameter Retrieval** [5]
    1. The process involves retrieving parameters [5]
    2. The parameter retrieval logic is currently **hard-coded** [9]
    3. It requires a **database connection** [10]
    4. The speaker admits to forgetting details about this part [13]

Extracting capture image...

1. **Ordering Strategy** [16]
    1. The speaker uses the **event name** for ordering [17]
    2. They are *not* using a message-based ordering system [18]

2. **Manual Numbering Implementation** [19]
    1. Previous ordering was already present via a number [19]
    2. The process involves prepending a number (one, two, three, etc.) to the event name [20]
    3. Each data iteration follows the same sequence [21]
    4. When the event is *initializing*, a 'one' is input [23]
    5. For a specific message, it receives 'two, three, four' [24]
    6. This ordering is **forced** by the speaker; Sumo does not do it automatically [25]

Extracting capture image...

1. **Testing the Order** [27]
    1. Running the process makes the ordering logic clear [27]

2. **Keyword Matching Flexibility** [28]
    1. One can use just a *space parameter* instead of the full text [28]
    2. This relies on **keyword matching** which still works [32]
    3. The speaker notes they are used to older methods [30]
    4. A caution is given to be safe with the bracelet (unclear context) [34]

Extracting capture image...

1. **All Events Panel** [35]
    1. The query pulls all messages for the **all events panel** [35]
    2. The speaker believes all events are covered by the Lambda execution [39]

2. **Failure Handling Process** [40]
    1. Failures are expected to occur eventually [40]
    2. During a failure, the event still arrives but *does not follow ordering* [42]
    3. After fixing the failure, the dashboard reflects the fix [43]
    4. Currently, there is no log data available for failure analysis [44]
    5. The speaker sometimes opens the code to check failure handling if time permits [45]

Extracting capture image...

1. **Current Event Count** [49]
    1. There are currently eight events visible [50]
    2. The database part of the process has just been completed [51]

2. **Panel Consolidation Discussion** [52]
    1. The speaker questions if the **all events panel** has too many events [52]
    2. They suggest removing parameters to shorten the list [54]
    3. Initial thought was to give parameters a *separate panel* [55]
    4. Removing the *retrieving parameter* is suggested [58]

3. **Database Query Sequence** [59]
    1. A need for a database for planning data is mentioned [59]
    2. The speaker notes the time is 3:45 already [62]
    3. The sequence includes: **Metric parameter**, **starting database query**, **executing database query** [63]
    4. All these steps are deemed necessary [64]
    5. The database pool connection creation is also required [78]
    6. The sequence concludes with: **Connection is successful**, **Query executed successfully**, **Query executed**, **Query completed** [79]
    7. Seven ordered events are confirmed present after this stage [86]

Extracting capture image...

1. **Next Steps: Couchbase Events** [87]
    1. Other events need to be seen and filtered [87]
    2. The next step involves **starting upload to Couchbase** [89]

2. **Handling Unmatched Events** [91]
    1. New events not matching existing keywords still appear [91]
    2. Their ordering might be slightly off (top or bottom) [92]
    3. Data will not be missed due to the handling logic [93]
    4. The logic either renames the event if the keyword matches [93]
    5. Or it passes the original event attribute value if no match occurs [94]

3. **Subsequent Couchbase Steps** [97]
    1. The process continues with **starting Couch processing data duplicates** [97]
    2. Duplicates are then **removed** [98]
    3. An event for *no data to upload* is noted [100]
    4. Processing planning data might be the ninth event if it occurs [107]
    5. The event *Processing planning data, duplicates removed* is considered very important [110]

Extracting capture image...

1. **Refactoring Suggestion** [113]
    1. It is suggested to combine all conditions into *one nested if* [113]
    2. The speaker notes they have never tried this structure [113]
    3. The proposed structure involves one `if` followed by multiple `event matches` on new lines [119]

2. **Syntax Clarification** [119]
    1. Questions arise about closing braces and indentation [129]
    2. The speaker confirms they have done this somewhere before but forgot [138]
    3. They request a picture of the screen to confirm syntax [121]

Extracting capture image...

1. **Problem with High Event Volume** [141]
    1. The main issue is that these systems have *too many events* [141]
    2. Renaming is necessary if an event matches *initializing* [145]
    3. The goal is to store different things in an event array [146]

2. **Alternative Sorting Methods** [149]
    1. Two ways to proceed are identified [149]
    2. **Approach 1**: Use the current method of adding a number for sorting [149]
    3. **Approach 2**: Just give a **blind count** [150]
    4. The current approach is considered the maximum possible effort [151]
    5. The speaker mentions they usually only handle about seven days of data [152]

3. **Nested IF Viability** [154]
    1. The speaker doubts the nested structure will work for this case [155]
    2. Nested logic works better for renaming *all* events or different panels [155]
    3. For every single event, the *correct number* must be added [155]
    4. The current structure is deemed quickly available [158]

Extracting capture image...

1. **Consultation Plan** [159]
    1. Another person is available to ask for advice [159]
    2. They plan to ask during the stand-up meeting tomorrow [162]
    3. The speaker prefers the current manual ordering method if possible [170]
    4. They are happy to list all events if the other person prefers that [166]

2. **Reviewing Past Complex Dashboard** [172]
    1. The speaker recalls working on a very tough dashboard previously [175]
    2. That dashboard was the *maddest thing* they worked on, but they learned much [176]
    3. They were a beginner when assigned that challenging task [178]
    4. They found a nested structure in an event called *process lambda events* [180]
    5. That structure was simpler, having only five to six events [181]
    6. The simpler structure used **batches** (B1, B2) for aggregation [182]
    7. Aggregating based on batch type made counting events easier [185]

3. **Next Panel Planning** [192]
    1. The current work will be handled later [193]
    2. The current panel is designated as the **third panel** for now [194]
    3. A large panel named *"All Events of the Usual Friend, CMDB added there"* will be added if time permits [195]

Extracting capture image...

1. **Error Panel Requirement** [197]
    1. The final planned panel is the **error panel** [197]
    2. They will use a **plain keyword match** for failures [202]
    3. The speaker confirms they have not intentionally caused failures [204]

2. **Error Count Event Discovery** [209]
    1. An existing *error count panel* was mentioned [210]
    2. The event *"Documents uploaded successfully"* is questioned as an event [211]
    3. This event logs the **error count** [213]
    4. The plan is to remove trash logs and **parse this error count** into a new panel [214]