Day - 9

14 August 2025 08:09

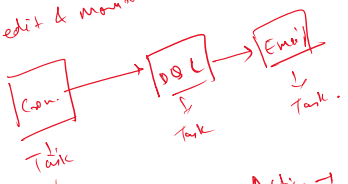
1. Workflows :-

Tool let you automatically act on monitoring data.

Pipeline sequence of activity with a condition.

① Repeated Activity in a certain seq.

② edit & monitor it



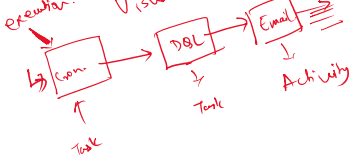
Action → Email, jira, ServiceNow

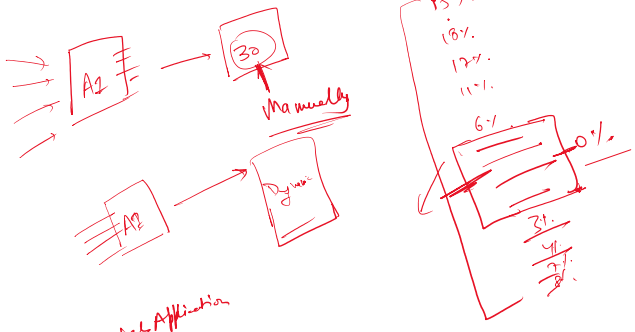
Execution :- ① triggered either by schedule or by event or manually.

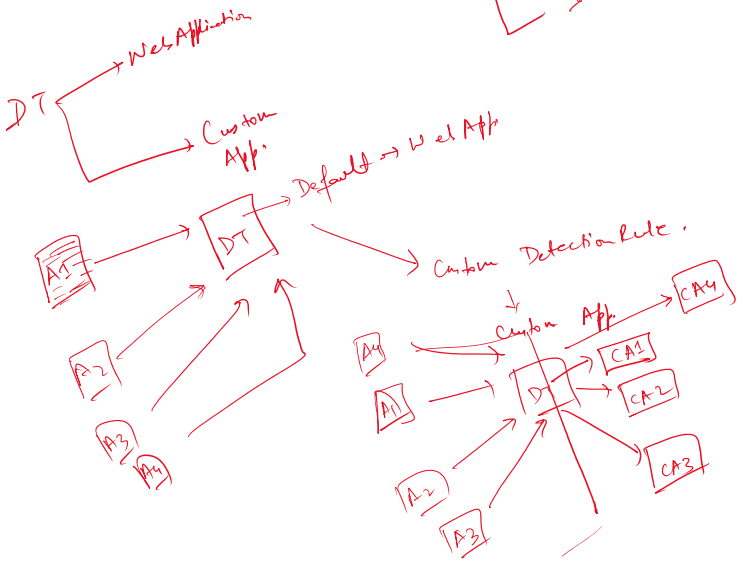
②

Schedule Notification →

Visualize the flow / intent







timeseries :- DSL Metric Command.

timeseries:- DQL Metric Command.

(1) starting of DQL.

(2) load, filter, & aggregate metric.

Syn:-

timeseries [column:] aggregation(metricKey[,filter:][,default:][,rollup:]
[,interval:|bin:]
[,rate:] [,interval:|bin:] - - - - -
[,from:] [,to:] - - -

rollup → min, max, sum, avg, total.
↓
time that should be used for the aggregation.

default:- fillgaps|empty bins → default =null

rate:- Duration that shall be used to adjust the bin values
(binValue|interval) * rate.

bin → how many chunks you want to split your time range
(12 to 1,500)

bucket_size = (total_time_range|bins)
↓
Nominal Time interval.

Timerange = 1hr, bins = 60
24hr → bins=24

JSON: arr | fieldAdd arr[ ]
?

Json Object:-

JSON: arr | fie

JSON{ —— }

JSON - extract all the field value

```
fetch logs, from:-5d
| filter log.source=="winevents.log"
| parse content, """JSON:content"""
| fields content[Severity],content[RecordNumber]
| summarize countDistinct(`content[RecordNumber]`),by:{`content[Severity]`}
//| summarize count(),by:{`content[Severity]`,`content[RecordNumber]`}
```

Json object extract a specific field.

```
fetch logs
| filter log.source=="winevents.log"
| parse content,"JSON{
LD:Severity
}:json_data"
| fieldsAdd severity=json_data[Severity]
| summarize severity_count=count(), by:{severity}
| append[fetch logs
| filter log.source=="winevents.log"
| parse content,"JSON{
INT : RecordNumber
}:json_data"
| fieldsAdd recordNumber = json_data[RecordNumber]
| summarize unique_record = countDistinct(recordNumber)]
```

Json Array:- - Parse any Json Array - Mixed type, nested stuff, - By default return variant_array (flexible, item can be of different data type) - typed array (all the item one type)

[5, null, 2.2, "17.9", "5\u0037\u0037\u0022"]

(5 → INT Double, 0; null → 1; 2.2 → Double, 2; "17.9" → string, 3; "5\u0037\u0037\u0022" → string, 4)

Json array: array

array[0] = 5 (INT)

array[1] = null

array[2] = 2.2 <Double>

array[3] = "17.9" <string>

array[4] = "___" <string>

["5", "null", "2.2", null, "null"]

JSON_array (typed=true){DOUBLE}:nums.

- JSON_array → DPL
- (typed=true) → force that the array has only one type of datatype
- {DOUBLE} → all the element in the array is double type
- nums → Variable | fieldName.

array has only one type of data type.

JSON-array (typed=true) {string}

1. JSON.array :arr → Variant array
2. JSON-array (typed=true) {Double} :num → Make it numerically.
3. {strict=false} → Be tolerant to messy JSON.
4. maxlen → very large arrays.

Openpipeline System:- Unified data pipeline, ingest, processing any data.

1. configure ingested,
2. Control access.
3. mark/secure → parse → DQL
   → Shards (-)
   SPL → (xxxx ...)

parse → enrich + transform + route.