Splunk to Kusto Query Language map

Article • 03/18/2023 • 6 minutes to read

This article is intended to assist users who are familiar with Splunk learn the Kusto Query Language to write log queries with Kusto. Direct comparisons are made between the two to highlight key differences and similarities, so you can build on your existing knowledge.

Structure and concepts

The following table compares concepts and data structures between Splunk and Kusto logs:

Concept	Splunk	Kusto	Comment
deployment unit	cluster	cluster	Kusto allows arbitrary cross-cluster queries. Splunk doesn't.
data caches	buckets	caching and retention policies	Controls the period and caching level for the data. This setting directly affects the performance of queries and the cost of the deployment.
logical partition of data	index	database	Allows logical separation of the data. Both implementations allow unions and joining across these partitions.
structured event metadata	N/A	table	Splunk doesn't expose the concept of event metadata to the search language. Kusto logs have the concept of a table, which has columns. Each event instance is mapped to a row.
data record	event	row	Terminology change only.
data record attribute	field	column	In Kusto, this setting is predefined as part of the table structure. In Splunk, each event has its own set of fields.
types	datatype	datatype	Kusto data types are more explicit because they're set on the columns. Both have the ability to work dynamically with data types and roughly equivalent set of datatypes, including JSON support.

Concept	Splunk	Kusto	Comment
query and search	search	query	Concepts essentially are the same between Kusto and Splunk.
event ingestion time	system time	`ingestion_time()`	In Splunk, each event gets a system timestamp of the time the event was indexed. In Kusto, you can define a policy called ingestion_time that exposes a system column that can be referenced through the ingestion_time() function.

Functions

The following table specifies functions in Kusto that are equivalent to Splunk functions.

Splunk	Kusto	Comment
`strcat`	`strcat()`	(1)
`split`	`split()`	(1)
`if`	`iff()`	(1)

`tonumber`	`todouble()`
	`tolong()`
	`toint()`
		(1)

`upper` `lower`	`toupper()` `tolower()`	(1)
`replace`	`replace_string()` or `replace_regex()`	(1) Although `replace` functions take three parameters in both products, the parameters are different.
`substr`	`substring()`	(1) Also note that Splunk uses one-based indices. Kusto notes zero-based indices.
`tolower`	`tolower()`	(1)
`toupper`	`toupper()`	(1)
`match`	`matches regex`	(2)
`regex`	`matches regex`	In Splunk, `regex` is an operator. In Kusto, it's a relational

Splunk	Kusto	Comment
		operator.
`searchmatch`	`==`	In Splunk, `searchmatch` allows searching for the exact string.
`random`	rand() rand(n)	Splunk's function returns a number between zero to 2^{31}-1. Kusto's returns a number between 0.0 and 1.0, or if a parameter is provided, between 0 and n-1.
`now`	`now()`	(1)
`relative_time`	`totimespan()`	(1) In Kusto, Splunk's equivalent of `relative_time(datetimeVal, offsetVal)` is `datetimeVal + totimespan(offsetVal)`. For example, `search | eval n=relative_time(now(), "-1d@d")` becomes `... | extend myTime = now() - totimespan("1d")`.

(1) In Splunk, the function is invoked by using the `eval` operator. In Kusto, it's used as part of `extend` or `project`.
(2) In Splunk, the function is invoked by using the `eval` operator. In Kusto, it can be used with the `where` operator.

Operators

The following sections give examples of how to use different operators in Splunk and Kusto.

> **Note**
>
> In the following examples, the Splunk field `rule` maps to a table in Kusto, and Splunk's default timestamp maps to the Logs Analytics `ingestion_time()` column.

Search

In Splunk, you can omit the `search` keyword and specify an unquoted string. In Kusto, you must start each query with `find`, an unquoted string is a column name, and the lookup

value must be a quoted string.

Product	Operator	Example
Splunk	`search`	`search Session.Id="c8894ffd-e684-43c9-9125-42adc25cd3fc" earliest=-24h`
Kusto	`find`	`find Session.Id=="c8894ffd-e684-43c9-9125-42adc25cd3fc" and ingestion_time()> ago(24h)`

Filter

Kusto log queries start from a tabular result set in which `filter` is applied. In Splunk, filtering is the default operation on the current index. You also can use the `where` operator in Splunk, but we don't recommend it.

Product	Operator	Example
Splunk	`search`	`Event.Rule="330009.2" Session.Id="c8894ffd-e684-43c9-9125-42adc25cd3fc" _indextime>-24h`
Kusto	`where`	`Office_Hub_OHubBGTaskError | where Session_Id == "c8894ffd-e684-43c9-9125-42adc25cd3fc" and ingestion_time() > ago(24h)`

Get *n* events or rows for inspection

Kusto log queries also support `take` as an alias to `limit`. In Splunk, if the results are ordered, `head` returns the first *n* results. In Kusto, `limit` isn't ordered, but it returns the first *n* rows that are found.

Product	Operator	Example
Splunk	`head`	`Event.Rule=330009.2 | head 100`
Kusto	`limit`	`Office_Hub_OHubBGTaskError | limit 100`

Get the first *n* events or rows ordered by a field or column

For the bottom results, in Splunk, you use `tail`. In Kusto, you can specify ordering direction by using `asc`.

Product	Operator	Example

Splunk	`head`	`Event.Rule="330009.2"`
		`| sort Event.Sequence`
		`| head 20`

Kusto	`top`	`Office_Hub_OHubBGTaskError` `| top 20 by Event_Sequence`

Extend the result set with new fields or columns

Splunk has an `eval` function, but it's not comparable to the `eval` operator in Kusto. Both the `eval` operator in Splunk and the `extend` operator in Kusto support only scalar functions and arithmetic operators.

Product	Operator	Example
Splunk	`eval`	`Event.Rule=330009.2` `| eval state= if(Data.Exception = "0", "success", "error")`
Kusto	`extend`	`Office_Hub_OHubBGTaskError` `| extend state = iff(Data_Exception == 0,"success" ,"error")`

Rename

Kusto uses the `project-rename` operator to rename a field. In the `project-rename` operator, a query can take advantage of any indexes that are prebuilt for a field. Splunk has a `rename` operator that does the same.

Product	Operator	Example
Splunk	`rename`	`Event.Rule=330009.2` `| rename Date.Exception as execption`
Kusto	`project-rename`	`Office_Hub_OHubBGTaskError` `| project-rename exception = Date_Exception`

Format results and projection

Splunk uses the `table` command to select which columns to include in the results. Kusto has a `project` operator that does the same and more.

Product	Operator	Example
Splunk	`table`	`Event.Rule=330009.2` `| table rule, state`
Kusto	`project`	`Office_Hub_OHubBGTaskError` `| project exception, state`

Splunk uses the `field -` command to select which columns to exclude from the results. Kusto has a `project-away` operator that does the same.

Product	Operator	Example
Splunk	`fields -`	`Event.Rule=330009.2` `| fields - quota, hightest_seller`
Kusto	`project-away`	`Office_Hub_OHubBGTaskError` `| project-away exception, state`

Aggregation

See the list of summarize aggregations functions that are available.

Splunk operator	Splunk example	Kusto operator	Kusto example
`stats`	`search (Rule=120502.*)` `| stats count by OSEnv, Audience`	`summarize`	`Office_Hub_OHubBGTaskError` `| summarize count() by App_Platform, Release_Audience`

`evenstats`	`...`
	`| stats count_i by time, category`
	`| eventstats sum(count_i) AS count_total by _time_`
		`join`	`T2`
			`| join kind=inner (T1) on _time`
			`| project _time, category, count_i, count_total`

Join

`join` in Splunk has substantial limitations. The subquery has a limit of 10,000 results (set in the deployment configuration file), and a limited number of join flavors are available.

Product	Operator	Example

Splunk	`join`	`Event.Rule=120103* &#124; stats by Client.Id, Data.Alias`
		`| join Client.Id max=0 [search earliest=-24h Event.Rule="150310.0"`
		`Data.Hresult=-2147221040]`

Kusto	`join`	`cluster("OAriaPPT").database("Office`
		`PowerPoint").Office_PowerPoint_PPT_Exceptions`
		`| where Data_Hresult== -2147221040`
		`| join kind = inner (Office_System_SystemHealthMetadata`
		`| summarize by Client_Id, Data_Alias)on Client_Id`

Sort

In Splunk, to sort in ascending order, you must use the `reverse` operator. Kusto also supports defining where to put nulls, either at the beginning or at the end.

Product	Operator	Example

Splunk	`sort`	`Event.Rule=120103`
		`| sort Data.Hresult`
		`| reverse`

Kusto	`order by`	`Office_Hub_OHubBGTaskError` `| order by Data_Hresult, desc`

Multivalue expand

The multivalue expand operator is similar in both Splunk and Kusto.

Product	Operator	Example
Splunk	`mvexpand`	`mvexpand solutions`
Kusto	`mv-expand`	`mv-expand solutions`

Result facets, interesting fields

In Log Analytics in the Azure portal, only the first column is exposed. All columns are available through the API.

Product	Operator	Example
Splunk	`fields`	`Event.Rule=330009.2` `| fields App.Version, App.Platform`
Kusto	`facets`	`Office_Excel_BI_PivotTableCreate` `| facet by App_Branch, App_Version`

Deduplicate

In Kusto, you can use `summarize arg_min()` to reverse the order of which record is chosen.

Product	Operator	Example
Splunk	`dedup`	`Event.Rule=330009.2` `| dedup device_id sortby -batterylife`
Kusto	`summarize arg_max()`	`Office_Excel_BI_PivotTableCreate` `| summarize arg_max(batterylife, *) by device_id`

Next steps

- Walk through a tutorial on the Kusto Query Language.

Feedback