A Deeper Dive into Flux

Grafanacon LA 2019

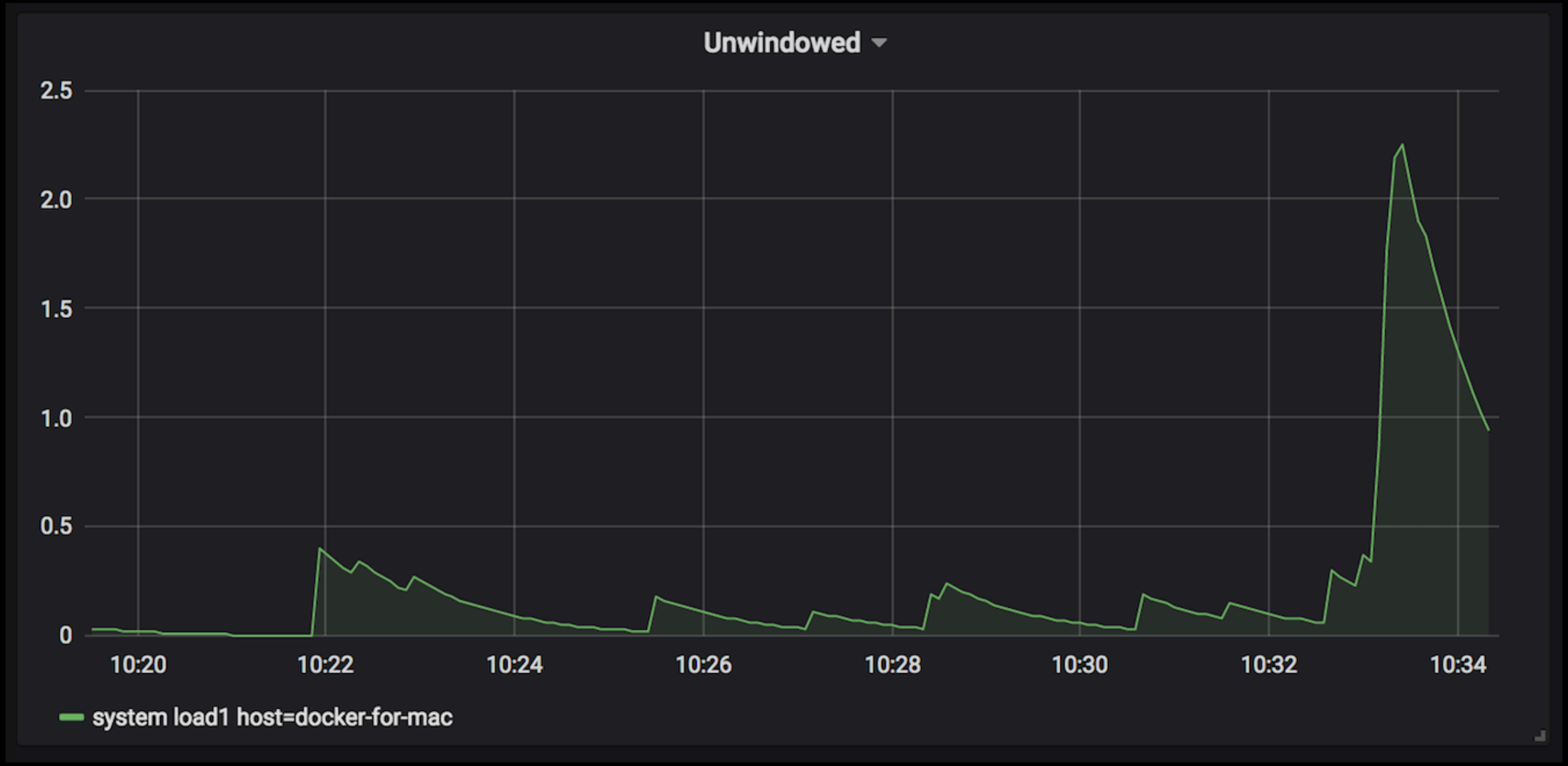
Flux

- Data Scripting Language
- Major component of InfluxDB 2.0 — will also be available standalone
- Functional language
- Built for querying, data exploration and munging

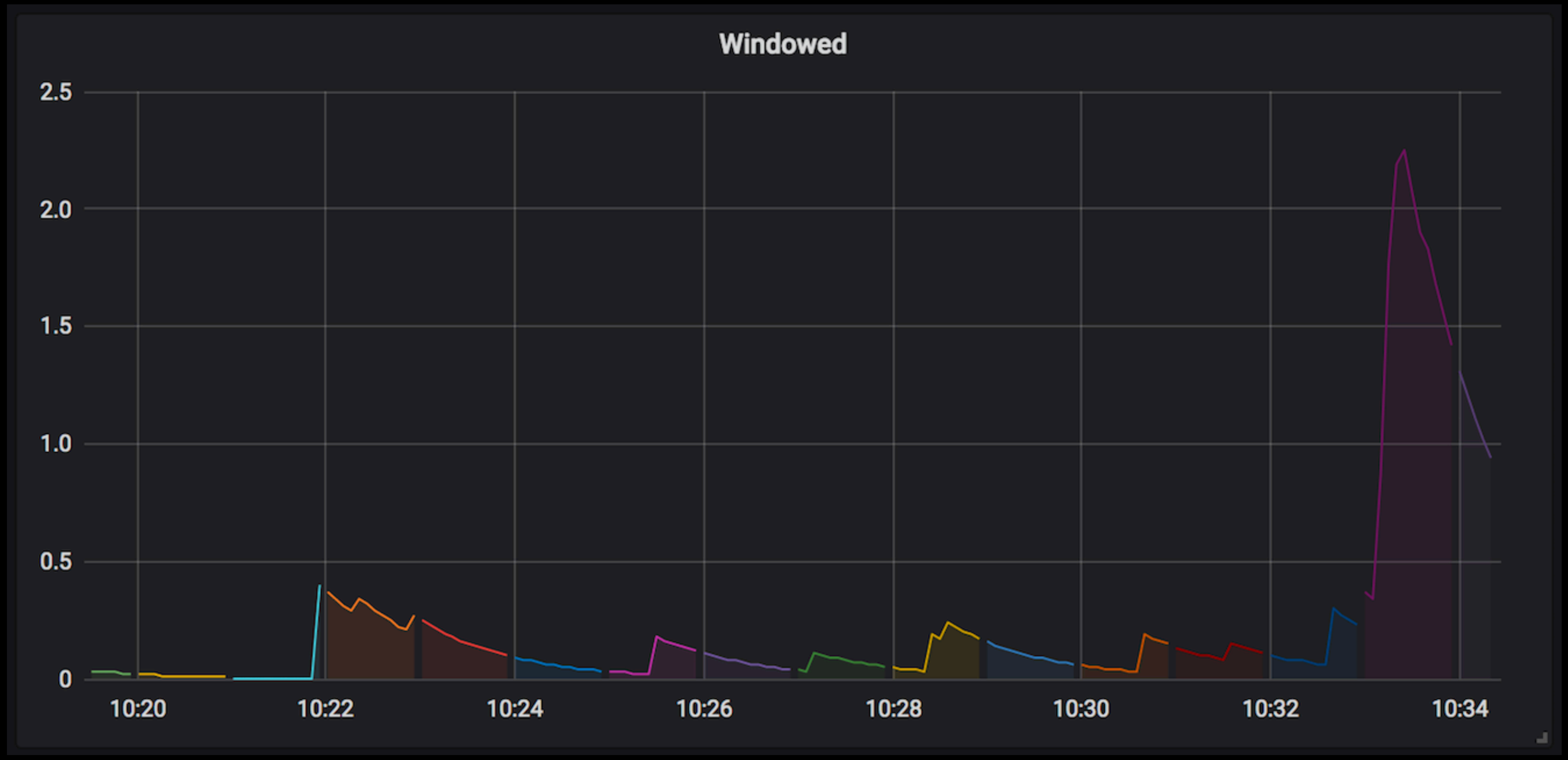
Features

- Windowing, Selectors, & Aggregates
- Grouping
- Joins
- Pivot

Windowing, Selectors, & Aggregates


Unwindowed
2.5
2.0
1.5
1.0
0.5
0
10:20
10:22
10:24
10:26
10:28
10:30
10:32
10:34
system load1 host=docker-for-mac

```
from(bucket:"telegraf/autogen")
    |> range(start:-15m)
    |> window(every:60s)
```


Windowed
2.5
2.0
1.5
1.0
0.5
0
10:20
10:22
10:24
10:26
10:28
10:30
10:32
10:34

Windowing Use Cases

- Align Data
- Smooth & transform data for analysis
- Work with irregular time series
- Downsample data

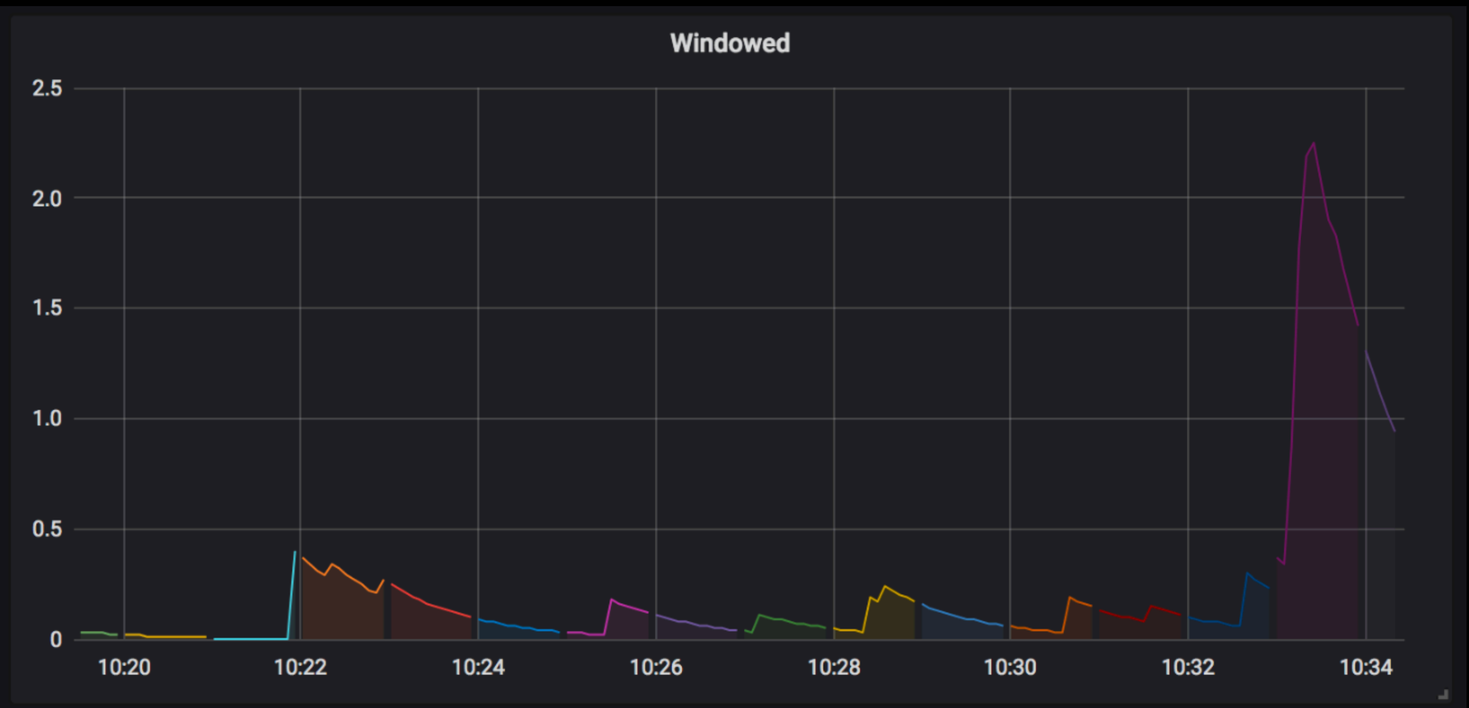
```
aggregateWindow(every: 1m, fn:mean)
```

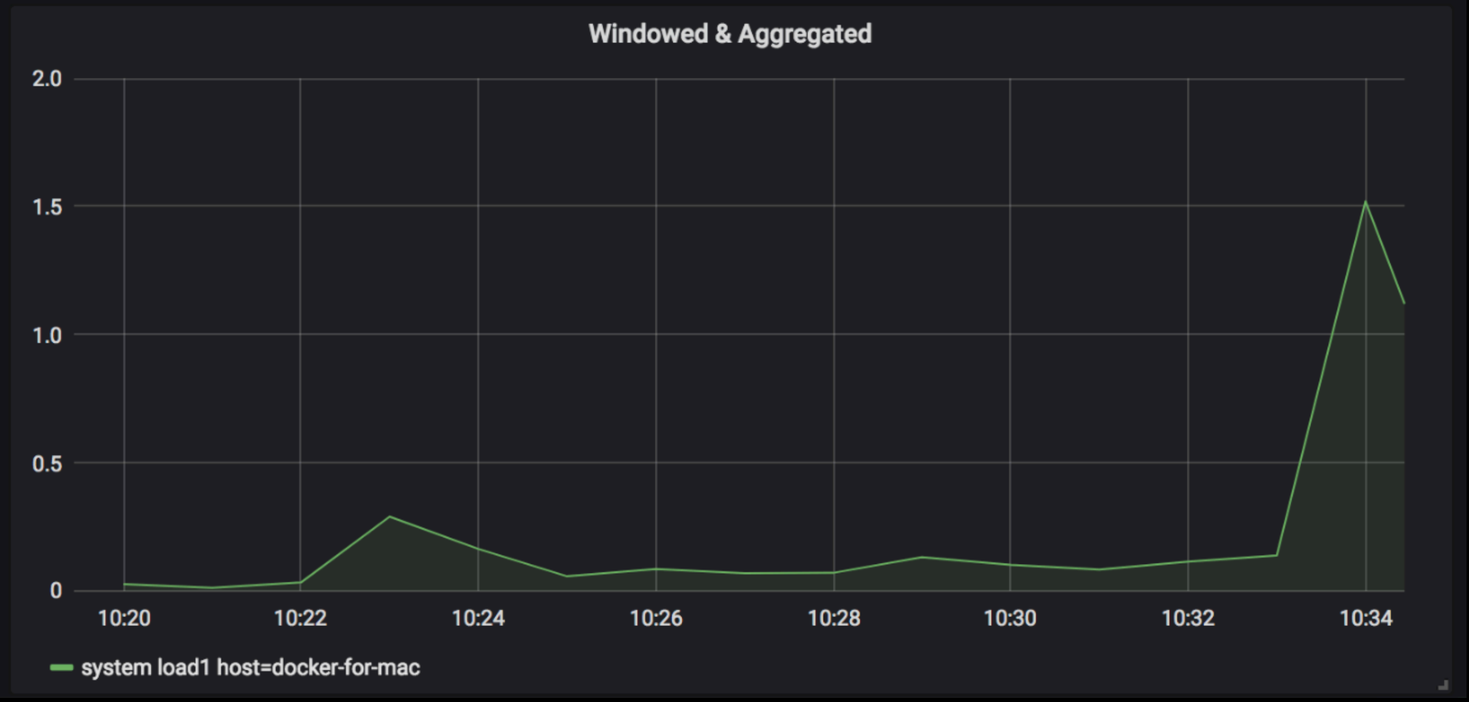
Aggregator Functions

- aggregateWindow
- count
- cov
- covariance
- derivative
- difference
- histogramQuantile
- increase
- integral
- mean
- median
- pearsonr
- percentile
- skew
- spread
- stddev
- sum
- highestAverage
- highestCurrent
- highestMax
- lowestAverage
- lowestCurrent
- lowestMin

Selector Functions

- bottom
- distinct
- first
- highestAverage
- highestCurrent
- highestMax
- last
- lowestAverage
- lowestCurrent
- lowestMin
- max
- min
- sample
- top
- unique


Windowed
2.5
2.0
1.5
1.0
0.5
0
10:20
10:22
10:24
10:26
10:28
10:30
10:32
10:34


Windowed & Aggregated
2.0
1.5
1.0
0.5
0
10:20
10:22
10:24
10:26
10:28
10:30
10:32
10:34
system load1 host=docker-for-mac

Grouping

Group Key

- Every table has a group key – a list of columns which for which every row in the table has the same value.
- For the “From” functions, the default group key for the tables is every column except _time and _value.

[_start, _stop, _field, _measurement, host]

```
> from(bucket: "telegraf/autogen") |> range(start: -5m) |> filter(fn: (r) => r._measurement == "system" and r._field == "load1") |>
drop(columns: ["_start", "_stop"]) |> limit(n: 5) |> yield()
Result: _result
Table: keys: [_field, _measurement, host]
        _field:string     _measurement:string           host:string                     _time:time                 _value:float
----------------------  ----------------------  ----------------------  ------------------------------  ---------------------------
                 load1                  system          docker-for-mac  2019-02-26T17:02:44.000000000Z                         0.11
                 load1                  system          docker-for-mac  2019-02-26T17:02:49.000000000Z                          0.1
                 load1                  system          docker-for-mac  2019-02-26T17:02:54.000000000Z                         0.09
                 load1                  system          docker-for-mac  2019-02-26T17:02:59.000000000Z                         0.08
                 load1                  system          docker-for-mac  2019-02-26T17:03:04.000000000Z                         0.16
Table: keys: [_field, _measurement, host]
        _field:string     _measurement:string           host:string                     _time:time                 _value:float
----------------------  ----------------------  ----------------------  ------------------------------  ---------------------------
                 load1                  system          noah-mbp.local  2019-02-26T17:02:43.000000000Z                         2.02
                 load1                  system          noah-mbp.local  2019-02-26T17:02:53.000000000Z                         1.93
                 load1                  system          noah-mbp.local  2019-02-26T17:03:03.000000000Z                         2.02
                 load1                  system          noah-mbp.local  2019-02-26T17:03:13.000000000Z                         2.01
                 load1                  system          noah-mbp.local  2019-02-26T17:03:23.000000000Z                         2.08
>
> from(bucket: "telegraf/autogen") |> range(start: -5m) |> filter(fn: (r) => r._measurement == "system" and r._field == "load1") |>
group(columns: ["_start", "_stop", "_field", "_measurement"]) |> drop(columns: ["_start", “_stop"]) |> yield()
Result: _result
Table: keys: [_field, _measurement]
        _field:string     _measurement:string                     _time:time                 _value:float           host:string
----------------------  ----------------------  ------------------------------  ---------------------------  ----------------------
                 load1                  system  2019-02-26T17:02:44.000000000Z                         0.11          docker-for-mac
                 load1                  system  2019-02-26T17:02:49.000000000Z                          0.1          docker-for-mac
                 load1                  system  2019-02-26T17:02:54.000000000Z                         0.09          docker-for-mac
  [...]
                 load1                  system  2019-02-26T17:03:13.000000000Z                         2.01          noah-mbp.local
                 load1                  system  2019-02-26T17:03:23.000000000Z                         2.08          noah-mbp.local
>
```

Grouping use case

- Extremely useful for “drilling down” into data
- Make comparisons
  - Example: Overall temperature of a system vs. temperature for individual components
- When using aggregators: Be explicit about your group keys

Joins

Join

- Highly requested feature
  - Supports: inner, cross, left, right, full (defaults to inner)
- Enables:
  - Math across measurements
  - With more “From” functions: addition of metadata from other sources
- Takes two arguments: tables an “on” columns to join on.
- Columns that must be renamed due to ambiguity (i.e. columns that occur in more than one input stream) are renamed according to the template <column>_<table>.

SF_Temperature

_time	_field	_value
1001	“temp”	70
1002	“temp”	75
1003	“temp”	72

NY_Temperature

_time	_field	_value
1001	“temp”	55
1002	“temp”	56
1003	“temp”	55

```
join(tables: {sf: SF_Temperature, ny: NY_Temperature}, on: ["_time", "_field"])
```

Output

_time	_field	_value_ny	_value_sf
1001	“temp”	55	70
1002	“temp”	56	75
1003	“temp”	55	72

```
httpd = from(bucket:"telegraf")
  |> range(start:start)
  |> filter(fn:(r) =>
    r._measurement == "influxdb_httpd" and
    r._field == "writeReq" and
    r.cluster_id == cluster_id
  )
  |> aggregateWindow(every: interval, fn: mean)
  |> derivative(nonNegative:true,unit:60s)

write = from(bucket:"telegraf")
  |> range(start:start)
  |> filter(fn:(r) =>
    r._measurement == "influxdb_write" and
    r._field == "pointReq" and
    r.cluster_id == cluster_id
  )
  |> aggregateWindow(every: interval, fn: max)
  |> derivative(nonNegative:true,unit:60s)

return join(
    tables:{httpd:httpd, write:write},
    on:["_time","_stop","_start","host"]
  )
  |> map(fn:(r) => ({
      _time: r._time,
      _value: r._value_httpd / r._value_write,
  }))
  |> group(columns: cluster_id)
```

Pivot

Pivot

- Pivot collects values stored vertically (column-wise) in a table and aligns them horizontally (row-wise) into logical sets.
- Good for re-shaping the data into a format suited for your application
- Use in combination with other functions to create summary tables
- Function takes three parameters:
  - rowKey, columnKey, valueColumn

Parameters

- RowKey is the list of columns which uniquely identify a row output
- ColumnKey is the list of columns used to pivot values onto each row identified by the rowKey
- ValueColumn identifies the single column that contains the value to be moved around the pivot

_time	_value	_measurement	_field
1970-01-01T00:00:00.000000001Z	1.0	"m1"	"f1"
1970-01-01T00:00:00.000000001Z	2.0	"m1"	"f2"
1970-01-01T00:00:00.000000001Z	null	"m1"	"f3"
1970-01-01T00:00:00.000000001Z	3.0	"m1"	null
1970-01-01T00:00:00.000000002Z	4.0	"m1"	"f1"
1970-01-01T00:00:00.000000002Z	5.0	"m1"	"f2"
null	6.0	"m1"	"f2"
1970-01-01T00:00:00.000000002Z	null	"m1"	"f3"
1970-01-01T00:00:00.000000003Z	null	"m1"	"f1"
1970-01-01T00:00:00.000000003Z	7.0	"m1"	null
1970-01-01T00:00:00.000000004Z	8.0	"m1"	"f3"

```
from(bucket:"test")
     |> range(start: 1970-01-01T00:00:00.000000000Z)
     |> pivot(rowKey:["_time"], columnKey: ["_field"], valueColumn: "_value")
```

_time	_measurement	f1	f2	f3	null
1970-01-01T00:00:00.000000001Z	"m1"	1.0	2.0	null	3.0
1970-01-01T00:00:00.000000002Z	"m1"	4.0	5.0	null	null
null	"m1"	null	6.0	null	null
1970-01-01T00:00:00.000000003Z	"m1"	null	null	null	7.0
1970-01-01T00:00:00.000000004Z	"m1"	null	null	8.0	null

Sonia Gupta’s Fabulous Pivot Blog

https://www.influxdata.com/blog/influxdb-reorganizing-data-with-the-pivot-function-in-flux/

community.influxdata.com

Feedback appreciated!

Twitter, GitHub: **@noahcrowley**

Email: **noah@influxdata.com**