Tetration

Cisco Tetration Analytics

Building better Dashboards v1

Ryan Tischer, CCIE 11459

Cisco

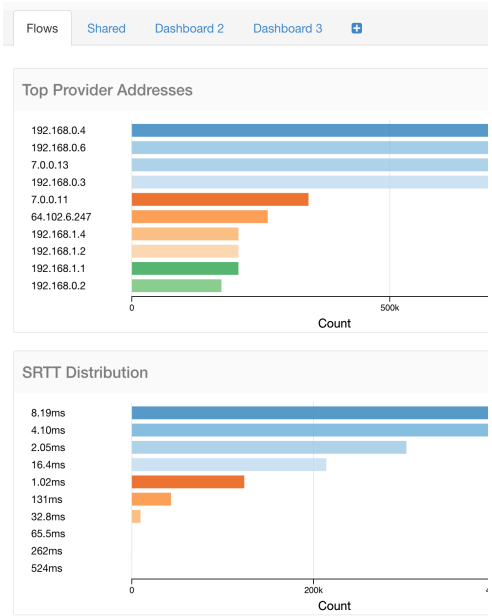
1/2017

TSA Call to Action

- Create interesting Dashboards are share.
- Better Demos that include Dashboards
- Need to figure out the best way to share? Github?
- Suggest we build a team to focus on TA Dashboards

Tetration Dashboards

- Summary of Tetration Data
- Designed for NOCs, Application owners, or C-level
- Can be broad i.e. total BW or based on specific data (App specific)
- Based on Druid



What is Druid

- Column-oriented, distributed data store
- Highly scalable (Petabytes)
- Supports real-time streams (Spark/Kafka)
- **Sub-second Queries**
- **Queries built in JSON formatted data**
- Used to power TA Dashboards and Flow Search

Example Data in Druid

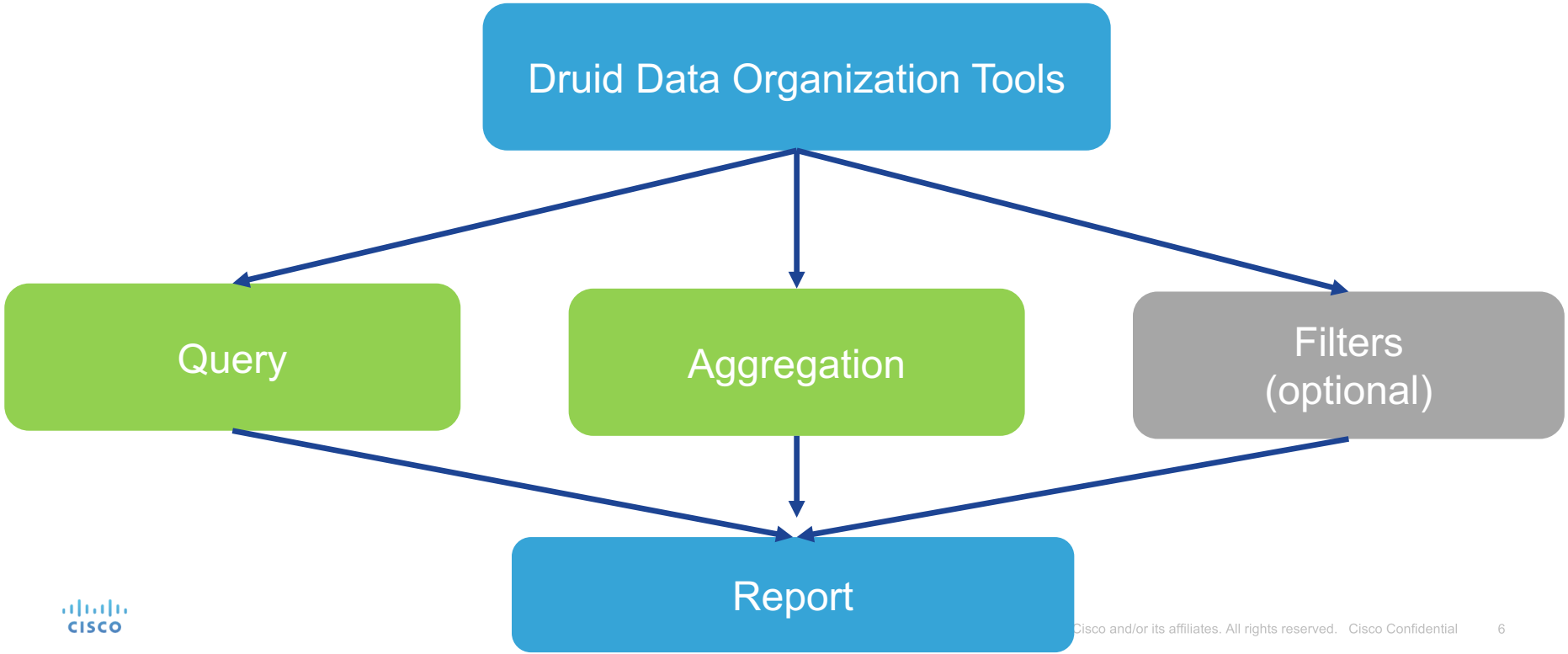
Wikipedia “edit” events

Timestamp	Page	Username	Gender	City	Characters Added	Characters Removed
2011-01-01T01:00:00Z	Justin Bieber	Boxer	Male	San Francisco	1800	25
2011-01-01T01:00:00Z	Justin Bieber	Reach	Male	Waterloo	2912	42
2011-01-01T02:00:00Z	Ke$ha	Helz	Male	Calgary	1953	17
2011-01-01T02:00:00Z	Ke$ha	Xeno	Male	Taiyuan	3194	170

Time stamped events in a table

Column-oriented

The Magic of Druid (TA Dashboards) is organizing data in away that answers questions.



Querying Data in Druid

- Three types of Queries
  - Timeseries
  - TopN
  - GroupBy
- First of two required inputs (Aggregation also required)
- Other query types are permitted, discussed later

Time series Query

- Group in blobs of time, formatted in JSON
- User set granularity (minute, hour, year, 15 minutes, etc)
- Druid returns a data point for each interval.
- For example a timeseries query by hour would produce

```
[ { "Count": 1511410,
"timestamp": "2017-01-16T17:00:00.000Z" },
 { "Count": 111644,
 "timestamp": "2017-01-16T18:00:00.000Z"} ]
```

The thing were counting

The timestamp (by hour)

Repeat for the next hour

The source of the Data is discussed later

TopN Query

- Sorted results of a data source, formatted in JSON
- Example Top talkers
- Druid returns a data point for each dimension (talker)
- For example top talkers by IP address

```
[ { "count": 73137,
"src_address”:"1.1.11.15",
"timestamp": "2017-01-16T17:03:23.000Z" },
{ "count": 72888,
"src_address": "1.1.11.10",
"timestamp": "2017-01-16T17:03:23.000Z" },
```

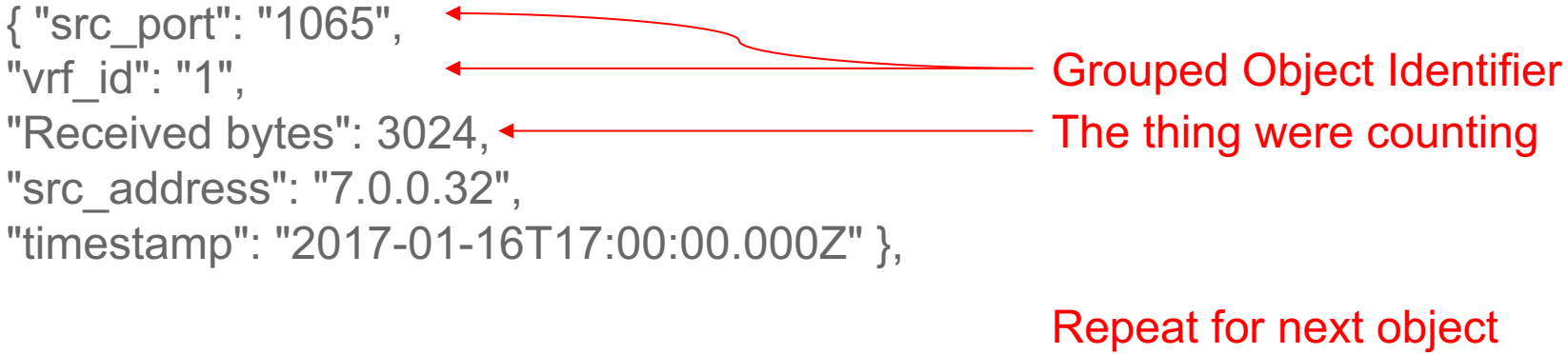
The thing were counting

Identifier

Repeat for next top talker

Groupby Query

- Multi-dimenstion query, formatted in JSON
- Similar to SQL “Group by”
- Druid returns a data point for each object
- For example source address + vrf + src port



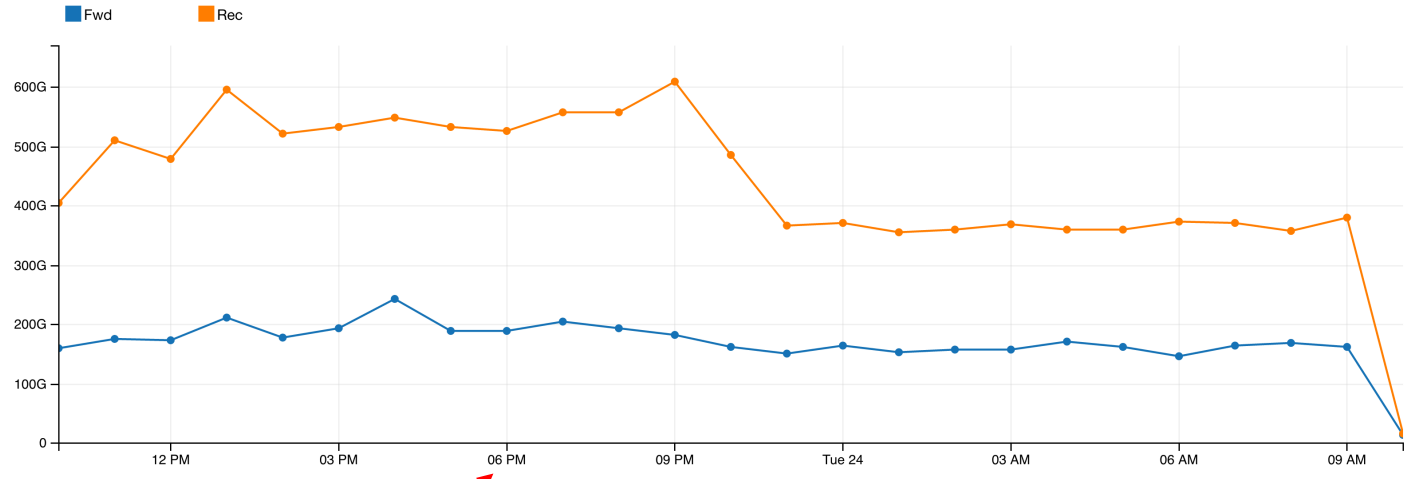
Aggregations

- Aggregate or group data into a single row
- Used with Queries
- For example
  - Query data in X timeseries, by hour and aggregate by Y types/metric
  - **Query** data from 1pm - 2pm by minutes and **aggregate** by sent bytes
- Metric includes count or any valid data within the dataset
- Multiple Aggregations Supported
  - Must be like data (sent/receive **bytes**) for the x axis to make sense

Query + Aggregations

- Queries build the X axis (per data source)
- Aggregations build the Y axis

Aggregation



Queries

Aggregations types / metrics

- Types
  - Count - Count the number of rows
  - DoubleSum – 64 bit integer value. Requires a metric
  - LongSum – 64 bit floating point value. Requires a metric. Not commonly used
  - Min/Max – Minimum or Maximum value in the set
  - Other are available, discussed later

Druid Filters

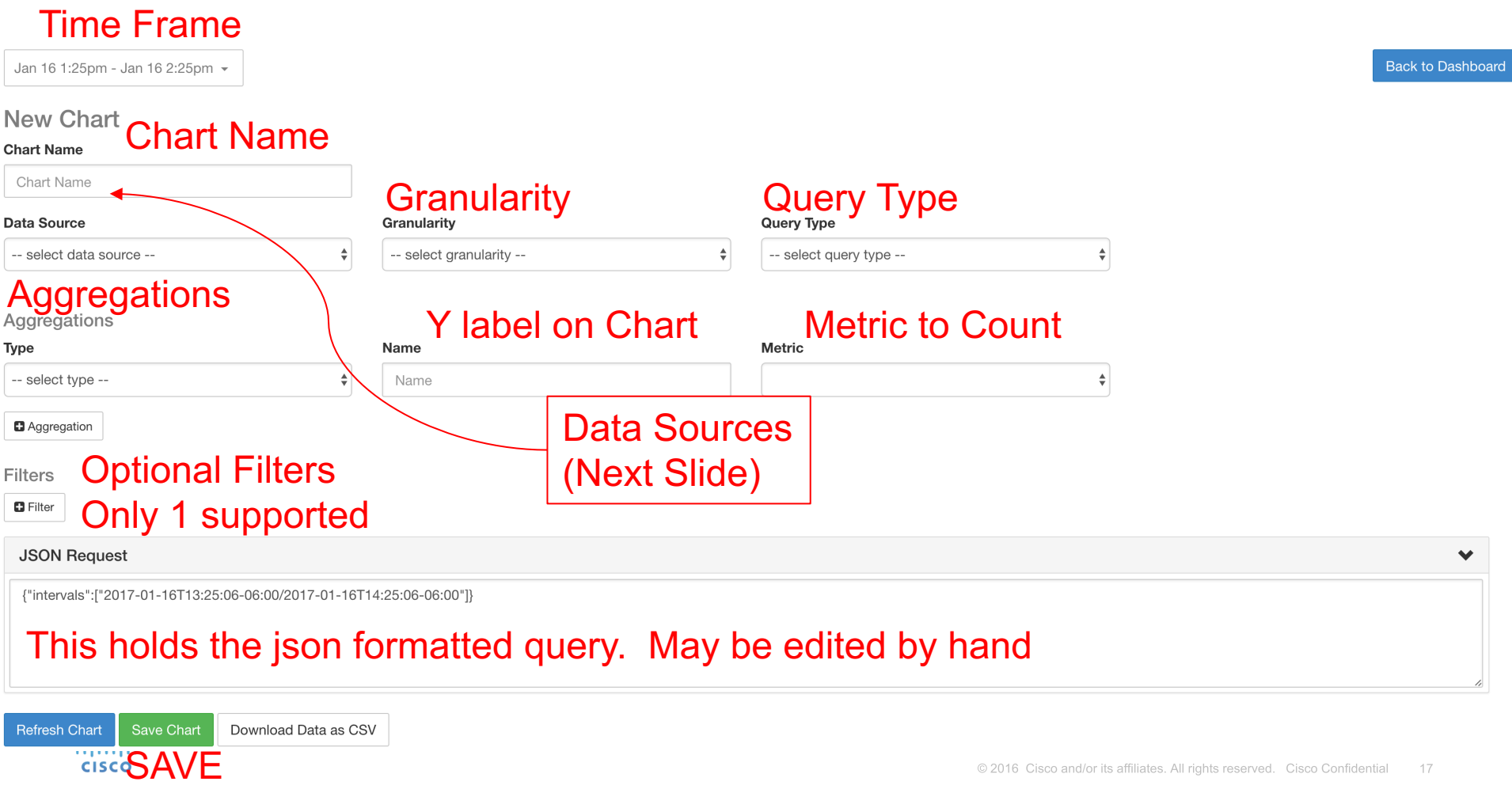
- Optional filter which rows are included in the dataset.
- Equivalent for SQL WHERE clause
- For example find rows where host name = SQL
- Two basic types of filters. Both are **CASE SENSITIVE**
  - Selector
    - Match a specific dimension. Hostname = App01
  - Regex
    - Any Regular expression (JAVA)
  - **Regex is preferred.**
- Other options discussed later

Regex Cheat Sheet

- "*" match zero or more characters
  - "App*" matches: App,"App1", "App2" ,"App01", "Appasfadsfasdfasab"
- "+" match one or more characters
  - App+ matches: ,"App1", "App2" ,"App01", "Appasfadsfasdfasab"
- (?i) before an expression makes the search **case-insensitive**
  - (?i)app01 matches: "app01", "apP01", "ApP01", "App01", etc
- [..] matches a range
  - App0[1-10] matches: "App01", "App09" etc
  - Ap[a-Z]01 matches: "Apt01", "App01", "ApP01", etc
- "^" is do not match
  - App0[^1] will not match App01

"$" signals end of string
App01$ matches: App01
(best practice)

Building Dashboards in Tetration



TA Data Sources

- Flow
- Hosts
- Attacks (no data yet)
- Policy Delta (no data yet)
- ADM runs appear as choices
  - However have no data (maybe future)
- Would be super nice if we had a dump of the raw table

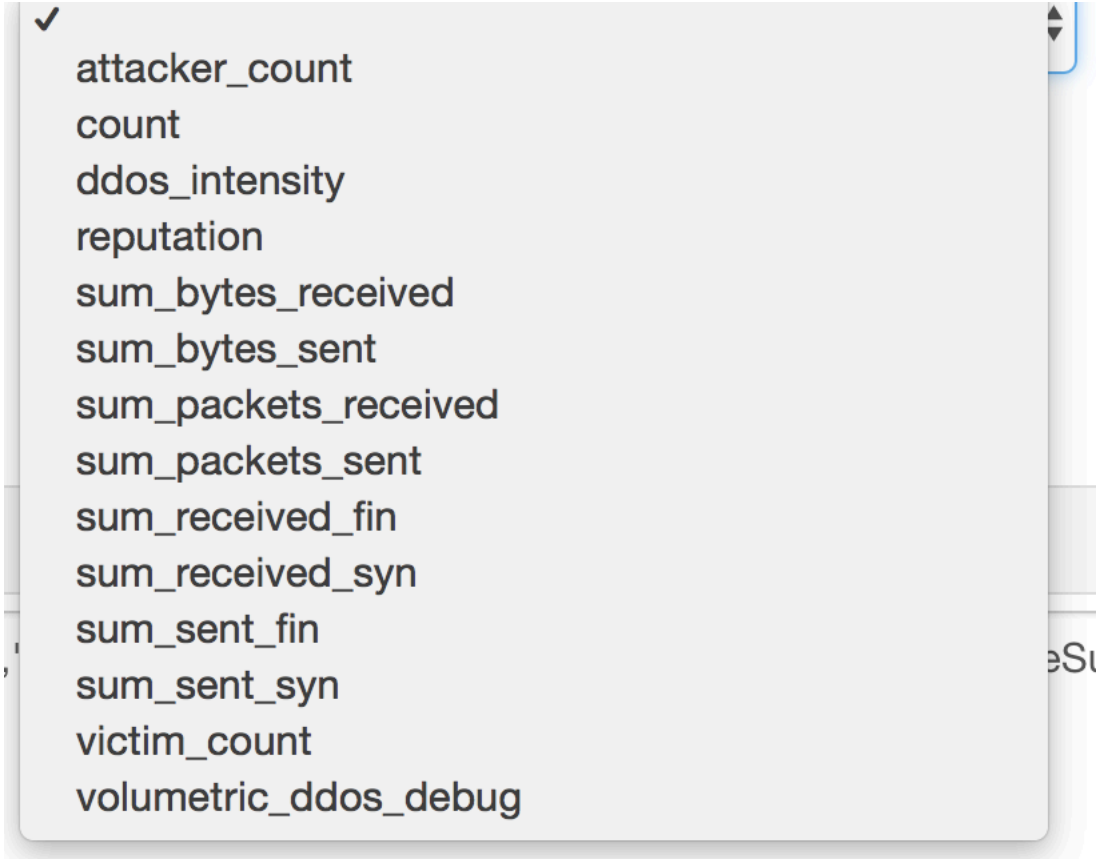
TA Data Source - Flow

- All flow data collected
- Exposed in flow search
- Metrics on any number
- Filter on anything
- Other data may be available


```
{
  "src_hostname": "web01",
  "src_address": "7.0.0.41",
  "src_port": "49167",
  "dst_hostname": "tet-13.internal",
  "dst_address": "172.26.46.35",
  "dst_port": "5640",
  "proto": "TCP",
  "key_type": "1",
  "start_timestamp": "1484512020000000",
  "rev_process_string": "/opt/tetration/collector/tet-collector --config_file /etc/tetration
    /collector/collector.config --timestamp_flow_info --logtostderr --max_num_ssl_sw_sensors 63000
    --enable_client_certificate true --write_empty_files true",
  "rev_process_owner": "tetter",
  "byte_count_dim": "16384",
  "pkt_count_dim": "32",
  "flow_duration_dim": "68719476736",
  "data_duration_dim": "68719476736",
  "srtt_dim_usec": "16384",
  "network_latency_dim_usec": "0",
  "server_app_latency_dim_usec": "0",
  "vrf_name": "Default",
  "fwd_policies": "MAYBE_PERMITTED:UNKNOWN",
  "rev_policies": "MAYBE_PERMITTED:UNKNOWN",
  "fwd_tags": "PSH ACK CWR",
  "rev_tags": "ACK",
  "record_start_ts_usec": 1484580180000000,
  "record_end_ts_usec": 1484580239372603,
  "vrf_id": "1",
  "count": 1,
  "srtt_usec": 16875,
  "total_network_latency_usec": 0,
  "server_app_latency_usec": 0,
  "fwd_pkts": 41,
  "rev_pkts": 41,
  "fwd_bytes": 19984,
  "rev_bytes": 2238,
  "fwd_flow_duration": 68219372603,
  "rev_flow_duration": 68219372593,
  "fwd_data_duration": 68219372603,
  "rev_data_duration": 0,
  "timestamp": "2017-01-16T15:23:00.000Z"
}
```

TA Data Source - Hosts

- All host data collected
- Other data may be available
- Aggregations →
- Filter on Host data
- All attacker/victim data is 0. Future release?



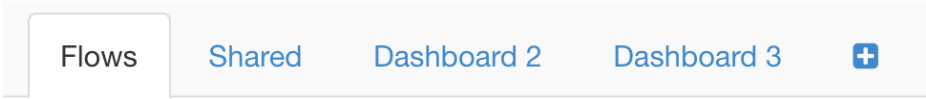
GUI → JSON

- The GUI builds the JSON request in real-time
- Can be edited by hand (more on this later)
- Used to share queries between dashboards or systems
- Use http://www.jsoneditoronline.org/ or other JSON editor to help visualize data
- Example →


```
{"dataSource":"flows",
"granularity":"all",
"queryType":"topN","threshold":10,"metric":"count",
"aggregations":[{"type":"count","fieldName":"","name":"count"}],
"intervals":["2017-01-16T13:25:06-06:00/2017-01-
16T14:25:06-06:00"],"dimension":"dst_port"}
```

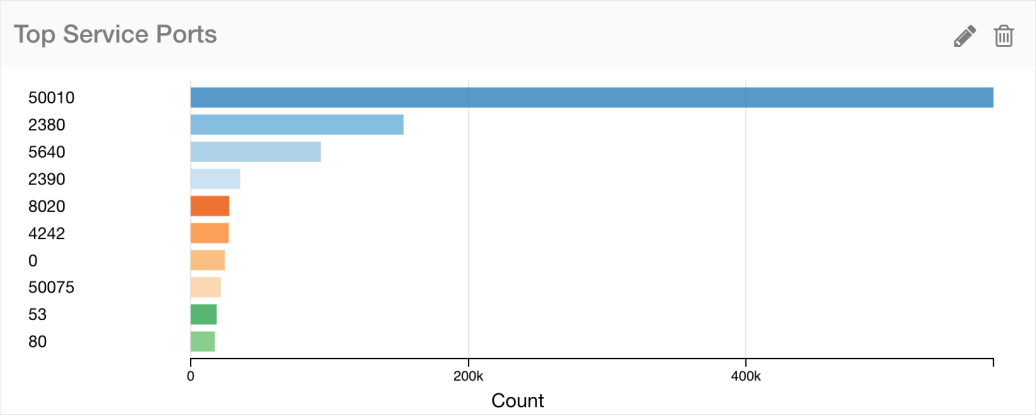
Dashboard hints

- Chart options will change based on…
  - Data Source
  - Query Type
  - Aggregations
- Use Chart preview and **JSON Response** to test data
  - If data count is all 0s you did something wrong
- TA Dashboard Tabs



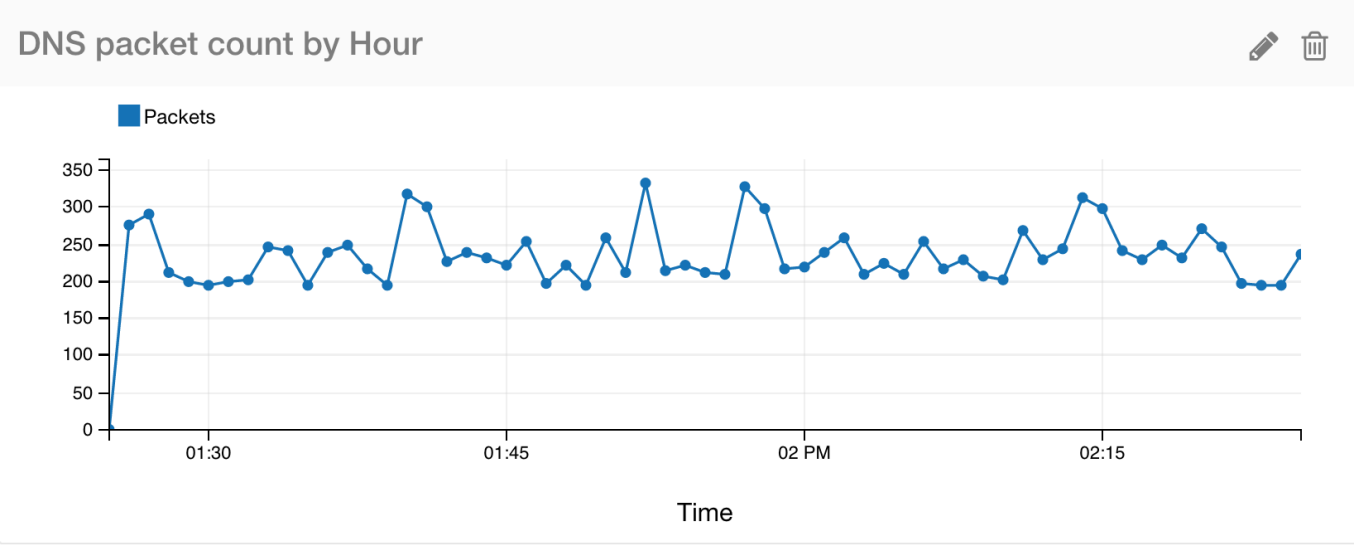
  - Flows tab is built in and can not be changed
  - Shared tab is shared between all users. Can be changed. Use as example
  - Users create personal dashboards

Dashboard Examples – Top Service Ports



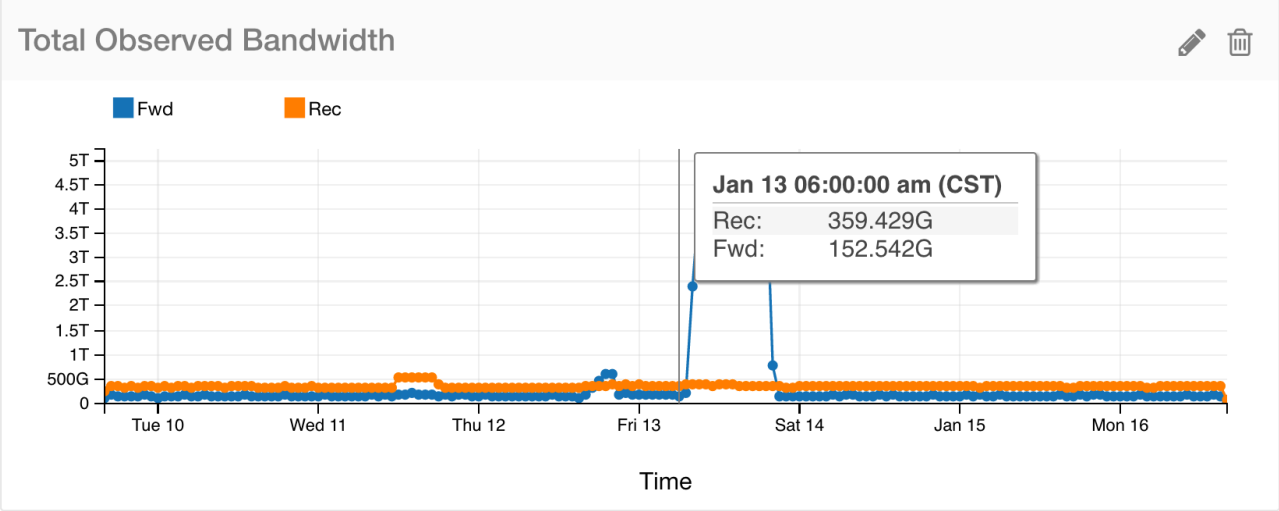
{"dataSource":"flows","granularity":"all","queryType":"topN","threshold":10,"metric":"count","aggregations":[{"type":"count","fieldName":"","name":"count"}],"intervals":["2017-01-16T13:25:06-06:00/2017-01-16T14:25:06-06:00"],"dimension":"dst_port"}

Dashboard Examples – DNS Packet Count



{"intervals":["2017-01-16T13:25:06-06:00/2017-01-16T14:25:06-06:00"],"dataSource":"flows","granularity":"minute","queryType":"timeseries","aggregations":[{"type":"doubleSum","fieldName":"rev_pkts","name":"Packets"}],"filter":{"type":"selector","dimension":"dst_port","value":"53"}}

Multiple Aggregations



Aggregations

Type	Name	Metric
doubleSum	Fwd	fwd_bytes
doubleSum	Rec	rev_bytes

TA Dashboard Off-Roading

Dashboard Off-roading

- Druid provides a rich query/visualization tools that are not exposed by the GUI
- Require editing JSON by hand
- A lot more work required!

Filters with AND / OR Expressions

- Druid Filters support AND/OR/NOT expressions
- Example…

Expression

Regular Filter

```
"filter": {
  "type": "and",
  "fields": [
    {
      "type": "regex",
      "dimension": "dst_hostname",
      "pattern": "[Ss][qQ][Ll]*"
    },
    {
      "type": "selector",
      "dimension": "dst_port",
      "value": "1433"
    }
  ],
  "dimension": null
}
}
```

Filters with NOT Expression

- Example…

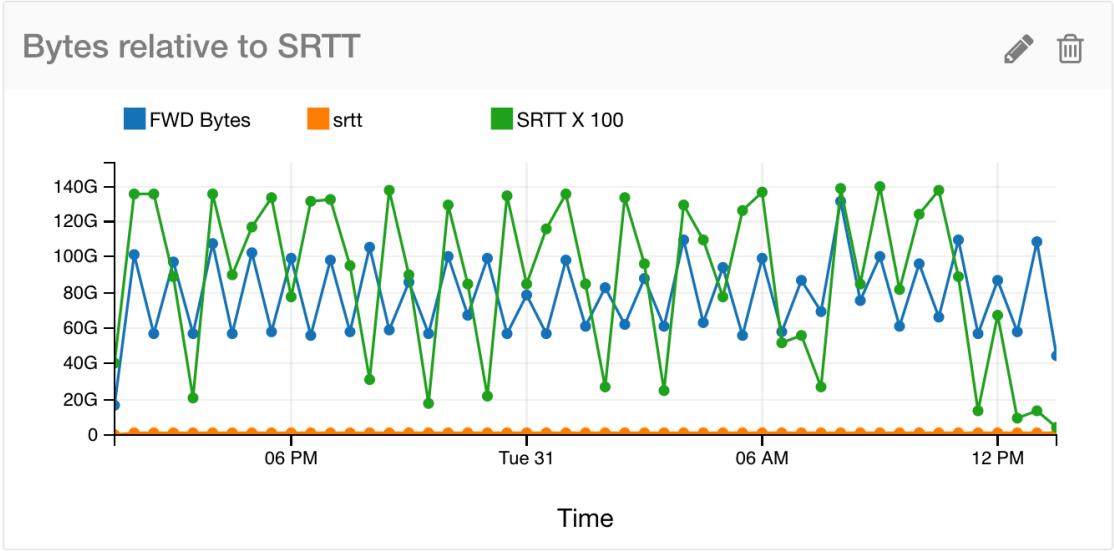
```
{
  "dataSource": "flows",
  "granularity": "all",
  "queryType": "topN",
  "threshold": 10,
  "metric": "count",
  "aggregations": [
    {
      "type": "count",
      "fieldName": "",
      "name": "count"
    }
  ],
  "intervals": [
    "2017-01-31T13:36:45-06:00/2017-01-31T14:36:45-06:00"
  ],
   "dimension": "proto",
  "filter": {
    "type": "not",
    "field":
      {
        "type": "regex",
        "dimension": "proto",
        "pattern": "TCP"
      }
  }
}
```

JavaScript Functions

- Javascript functions serve as aggregators, filters, or post-aggregations. **Must return a number**
- Useful for complex manipulation
- Druid documentation is terrible and no one seems to use this
- JS functions will provide much better data and add context
- Examples include
  - Ratio of traffic behind load balancers
  - Total resources consumed for a multitier application
  - Port BW consumption / total
  - TCP sessions / server capacity ← app scaling

JavaScript Aggregator Example

```
{
  "intervals": [
    "2017-01-30T13:50:05-06:00/2017-01-31T13:50:05-06:00"
  ],
  "dataSource": "flows",
  "granularity": "thirty_minute",
  "queryType": "timeseries",
  "aggregations": [
    {
      "type": "doubleSum",
      "name": "FWD Bytes",
      "fieldName": "fwd_bytes"
    },
    {
      "type": "doubleSum",
      "name": "srtt",
      "fieldName": "srtt_usec"
    },
    {
      "type": "javascript",
      "name": "SRTT X 100",
      "fieldNames": [
        "srtt_usec"
      ],
      "fnAggregate": "function(current, a)       { return current +  (a * 100); }",
      "fnCombine": "function(partialA)  { return partialA; }",
      "fnReset": "function()            { return 0; }"
    }
  ]
}
```



JavaScript Aggregator Example

```
{
  "intervals": [
    "2017-01-30T13:50:05-06:00/2017-01-31T13:50:05-06:00"
  ],
  "dataSource": "flows",
  "granularity": "thirty_minute",
  "queryType": "timeseries",
  "aggregations": [
    {
      "type": "doubleSum",
      "name": "FWD Bytes",
      "fieldName": "fwd_bytes"
    },
    {
      "type": "doubleSum",
      "name": "srtt",
      "fieldName": "srtt_usec"
    },
    {
      "type": "javascript",
      "name": "SRTT X 100",
      "fieldNames": [
        "srtt_usec"
      ],
      "fnAggregate": "function(current, a)       { return current +  (a * 100); }",
      "fnCombine": "function(partialA)  { return partialA; }",
      "fnReset": "function()            { return 0; }"
    }
  ]
}
```

All 3 functions must be present

fnAggregate

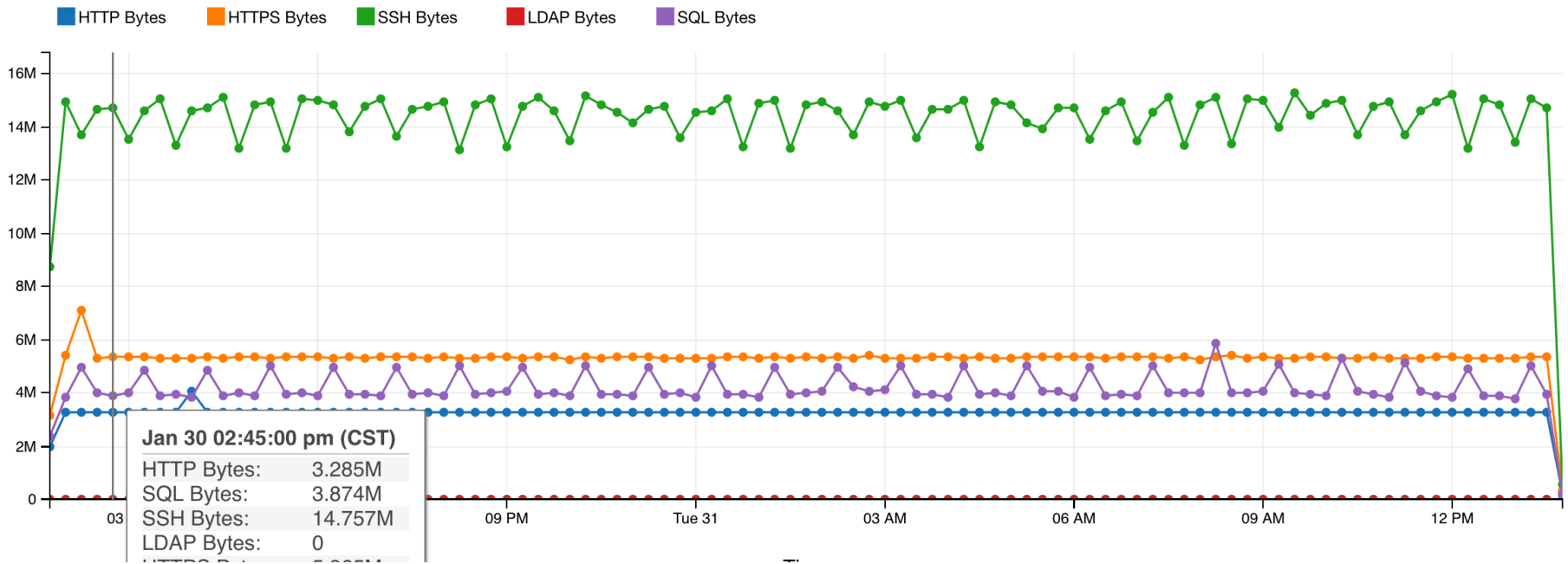
- "current" – unclear why
- "a" – ordered list of from "fieldnames" list
- **return** the result

fnCombine - aggregate the different rows

fnReset – init value

Filtered Aggregator

- Filter aggregator on the fly

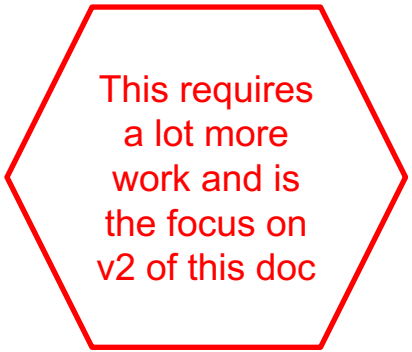


Filtered Aggregator

```
{
  "intervals": [
    "2017-01-30T13:50:05-06:00/2017-01-31T13:50:05-06:00"
  ],
  "dataSource": "flows",
  "queryType": "timeseries",
  "granularity": "fifteen_minute",
  "aggregations": [
    {
      "type": "filtered",
      "name": "HTTP Bytes",
      "filter": {
        "type": "selector",
        "dimension": "dst_port",
        "value": "80"
      },
      "aggregator": {
        "type": "doubleSum",
        "fieldName": "rev_bytes",
        "name": "HTTP Bytes"
      }
    },
    {
      "type": "filtered",
      "name": "HTTPS Bytes",
      "filter": {
        "type": "selector",
        "dimension": "dst_port",
        "value": "443"
      },
      "aggregator": {
        "type": "doubleSum",
        "fieldName": "rev_bytes",
        "name": "HTTPS Bytes"
      }
    },
    {
      "type": "filtered",
      "name": "SSH Bytes",
      "filter": {
        "type": "selector",
        "dimension": "dst_port",
        "value": "22"
```

Post Aggregations

- TA seems to ignore input



Cardinality Aggregations

- Determine the number of distinct X
- This may be useful for
  - Number of servers with port 80 open
  - Number of different servers talking to SQL
  - Distinct EPGs