Splunk Intermediate/Advanced

Course Outline

Module 1: Course Introduction
Module 2: Beyond Search Fundamentals
Module 3: Commands for Visualizations
Module 4: Advanced Visualizations
Module 5: Filtering and Formatting Data
Module 6: Correlating Events
Module 7: Introduction to Knowledge Objects
Module 8: Creating and Managing Fields
Module 9: Creating Field Aliases and Calculated Fields
Module 10: Creating Tags and Event Types
Module 11: Creating and Using Macros
Module 12: Creating and Using Workflow Actions
Module 13: Creating Data Models
Module 14: Using the Common Information Model (CIM) Add-On

Buttercup Games, Inc.

- Multinational company with HQ in San Francisco and offices in Boston and London
- Sells product mainly through its worldwide chain of third party stores, but also sells through its online store

Your Role at Buttercup Games

- You are a Splunk power user with a great understanding of all your company's data
- Your responsibilities are to provide information to users throughout the company and to create and manage Splunk knowledge objects for your stakeholders
- You implement best practices for naming conventions of all knowledge objects
- You gather data and statistics, and report on Security, IT Operations, Operational Intelligence, etc.

Review Search Fundamentals

Basic Search Review

- **Keywords**

  For example, search for a single word (e.g., `error`) or group of words (e.g., `error password`)

- **Booleans**

  NOT, OR, AND; AND is implied; MUST be uppercase; can use ( )'s to force precedence

  `sourcetype=vendor_sales OR (sourcetype=access_combined action=purchase)`

- **Phrases**

  `"web error"` (different than `web AND error`)

- **Field searches**

  `status=404, user=admin`

- **Wildcard (*)**

  - `status=40*` matches 40, 40a, 404, etc.
  - Starting keywords with a wildcard is very inefficient, e.g. ***dmin**

- **Comparisons**

  `=, !=, <, <=, >=, > status>399, user!=admin`

Basic Search Review (cont.)

- `table`: returns table containing only specified fields in result set
- `rename`: renames a field in results
- `fields`: includes or excludes specified fields
- `dedup`: removes duplicates from results
- `sort`: sorts results by specified field
- `lookup`: adds field values from external source (e.g., csv files)

Case Sensitivity – Sensitive

Case sensitive	Examples
Boolean operators (uppercase)	AND, OR, NOT (Boolean operators) and, or, not (literal keywords)
Field names	`productId` vs. `productid` `eval cs_username = "Total Access"`
Field values from lookup (default, but configurable)	`product_name="Tulip Bouquet"` vs. `product_name="tulip bouquet"`
Regular expressions	`\d\d\d` vs. `\D\D\D`

eval and where commands	`eval action=if(action=="view",...)`
	`where action="Purchase"`
	`stats count(eval(action="view")) as...`

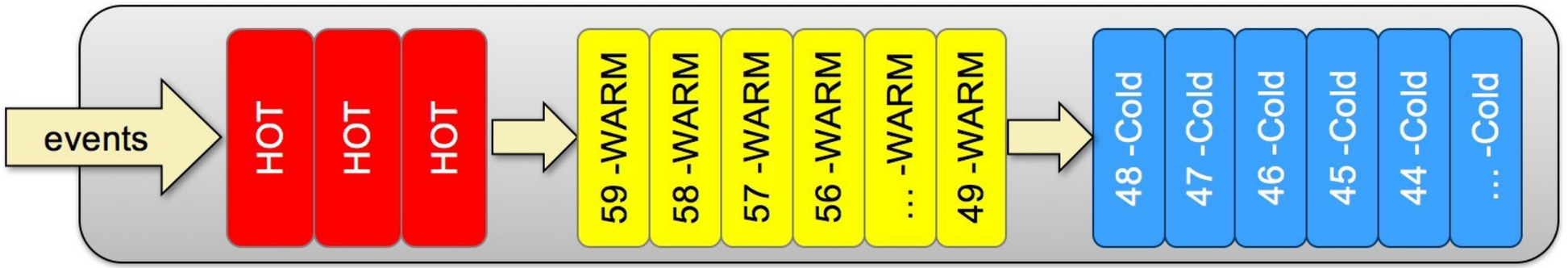
Tags	tag=DMZ vs. tag=dmz

Case Sensitivity – Insensitive

Case insensitive	Examples
Command names	STATS, stats, sTaTs
Command clauses	AS used by stats, rename, ...; BY used by stats, chart, top, ...; WITH used by replace
Search terms	failed, FAILED, Failed
Statistical functions	avg, AVG, Avg used by stats, chart, ...
Field values	host=www1, host=WWW1 (unless coming from a lookup)

How Splunk Searches – Buckets

- As events come in, Splunk places them into an index's hot bucket (only writable bucket)
- As buckets age, they roll from the hot to warm to cold
- Each bucket has its own raw data, metadata, and index files
- Metadata files track source, sourcetype, and host
- Admins can add more



How Splunk Searches – Searching

- When you search, Splunk uses the time range to choose which buckets to search and then uses the bucket indexes to find qualifying events
- When you search for `index=web password fail*` during the last 24 hours:
  - Splunk identifies the buckets for the last 24 hours
    - And searches the indexes of those buckets for the search terms

Hot: Now to -3h	index	raw events
Hot: -3 to -6h	index	raw events
Hot: -5 to -8h	index	raw events
Warm: -9 to -12h	index	raw events
Warm: -12 to -15h	index	raw events
Warm: -14 to -17h	index	raw events
.....	index	raw events
Warm: -42 to -45h	index	raw events
Warm: -45 to -48h	index	raw events
Cold: -48 to -51h	index	raw events
Cold: -51 to -54h	index	raw events
Cold: -54 to -57h...	index	raw events

General Search Practices

- As events are stored by time, time is the most efficient filter
- After time, most powerful keywords are host, source, sourcetype
- To make searches more efficient, include as many terms as possible
  - e.g., searching for `sourcetype=x failure` is better than `failure`
- Use the fields command to extract (discover) only fields you need
- Example: Search last 365 days, scans 566,720 events (in secs):

```
index=web sourcetype=access_combined                                  15.16
index=web sourcetype=access_combined |fields clientip bytes referrer   4.49
```

General Search Practices – Wildcards

- Splunk only searches for whole words, but wildcards allowed
- Only *trailing* wildcards make efficient use of index
  - Wildcards at *beginning* of string scan all events within time frame
  - Wildcards in *middle* of string may return inconsistent results
  - So use `fail*` (not `*fail` or `*fail*` or `f*il`)
- Wildcards tested after all other terms

General Search Practices

- Inclusion is generally better than exclusion
  - Searching for "access denied" is faster than NOT "access granted"
- Filter as early in your search as possible
  - Removing duplicates then sorting is faster than sorting then removing duplicates
- Use the appropriate search mode
  - Fast - performance over completeness
  - Smart [default]
  - Verbose - completeness over performance

Transforming Search Commands

- A transforming command:
  - Massages raw data into a data table
  - 'Transforms' specified cell values for each event into numerical values that you can use for statistical purposes
  - Is required to 'transform' search results into visualizations
- Transforming commands include:
  - top
  - rare
  - chart
  - timechart
  - stats
  - geostats

Reviewing Search Mode – Fast Mode

- Emphasizes performance, returning only essential and required data
- For non-transforming searches:
  - ✓ Events – fields sidebar displays only those fields required for the search
  - ✓ Patterns
- Contents of interesting fields sidebar are lost

Reviewing Search Mode – Smart Mode (Default)

- Designed to give you the best results for your search
- Combination of Fast and Verbose modes
- For non-transforming searches:
  - ✓ Events – fields sidebar displays all fields
  - ✓ Patterns

- For transforming searches:
  - ✓ Statistics or visualizations

Reviewing Search Mode – Verbose Mode

- Emphasizes completeness by returning all possible field and event data
- For non-transforming searches:
  - ✓ Events – fields sidebar displays all fields
  - ✓ Patterns

- For transforming searches:
  - ✓ Events
  - ✓ Patterns
  - ✓ Statistics or visualizations

Search Performance – Modes

- Use the most appropriate search mode:

```
index=web
sourcetype=access_combined
| chart count by product_name
```

- Time range: last 365 days

	Returned Results	Events Scanned	Time
	14	566,731	1.82
Smart	14	566,731	1.91
Verbose	14	566,731	15.21

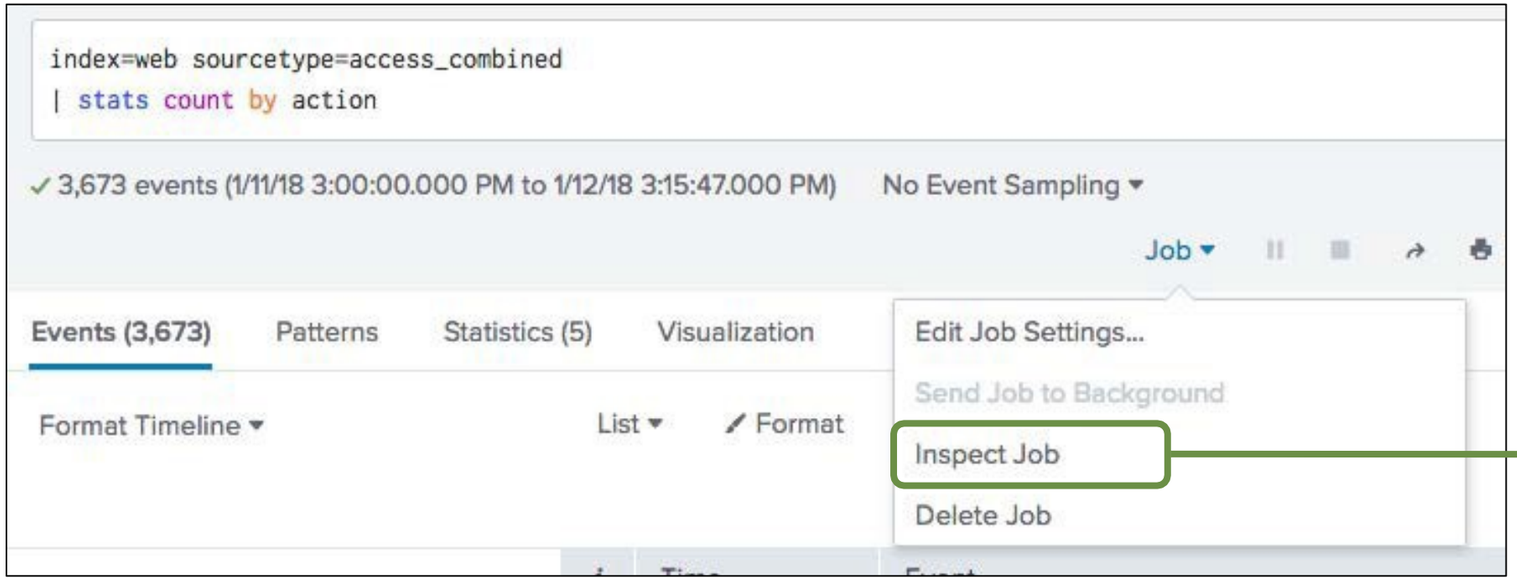
Search Performance – Types of Searches

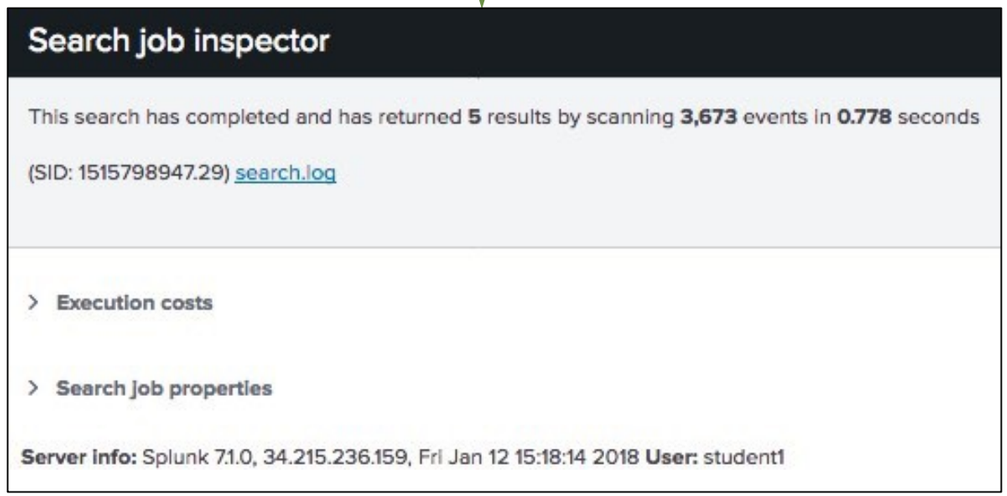
Search	Description	Indexer throughput
Dense	A large percentage of the data matches the search Use Cases: computing stats, reporting `index=web sourcetype=access_combined | timechart count`	Up to 50K matching EPS (Events per second) *CPU bound*
Sparse	A small percentage of data matches the search Use Cases: troubleshooting, error analysis `index=web sourcetype=access_combined status=404 | timechart count`	Up to 5K matching EPS *CPU bound*
Super Sparse	Returns a small number of results from each index bucket matching the search I/O intensive as the indexer looks through all of an index's buckets With a lot of data, with a lot of buckets, it can take a long time to finish `index=web sourcetype=access_combined action=denied src_ip=10.2.3.11`	Up to 2 seconds per index bucket *I/O bound*
Rare	The indexer checks all buckets to find results, but bloom filters eliminate those buckets that don't include search results Use Cases: user behavior tracking `index=web sourcetype=access_combined sessionID=1234`	Up to 10-50 index buckets/ second *I/O bound*

Search Job Inspector

- Tool allows you to examine:
  - Overall stats of search (e.g., records processed and returned, processing time)
  - How search was processed
  - Where Splunk spent its time
- Use to troubleshoot search's performance and understand impact of knowledge objects on processing (e.g., event types, tags, lookups)
- Any existing (i.e., not expired) search job can be inspected

Search Job Inspector – 3 Components

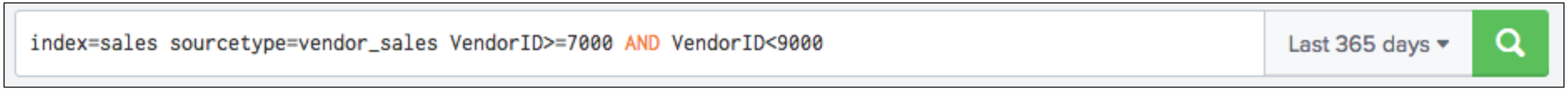




- Header
- Execution costs
- Search job properties

Search Job Inspector – Search Job Properties

Example:



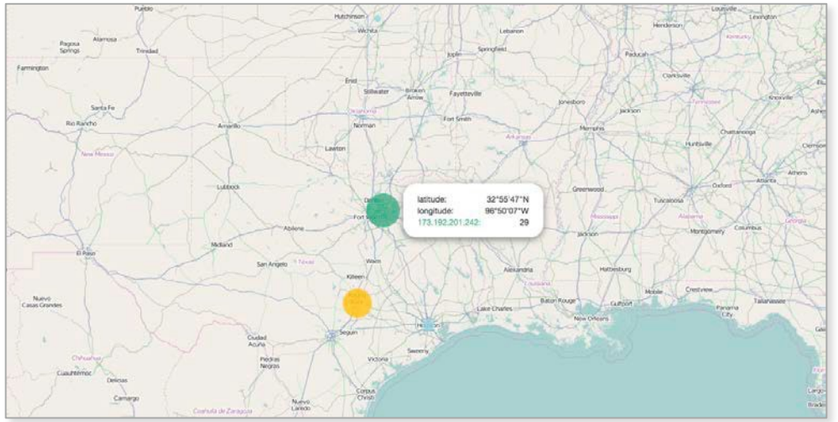
- Produces scanCount of 127,201 events
- Returns resultCount of 2,144 in 3.01 seconds
- To calculate performance:
  - Do **not** use resultCount/time 2,144 / 3.01 = 712 EPS*
  - Rather, calculate scanCount/time 127,201/ 3.01 = 40,892 EPS

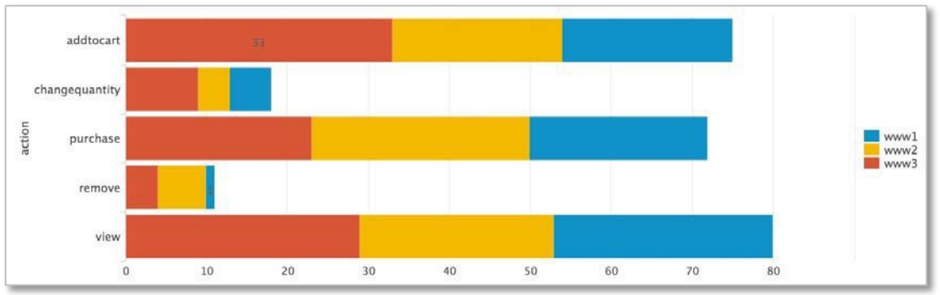
* EPS= events per second

Commands for Visualizations

Visualization Types

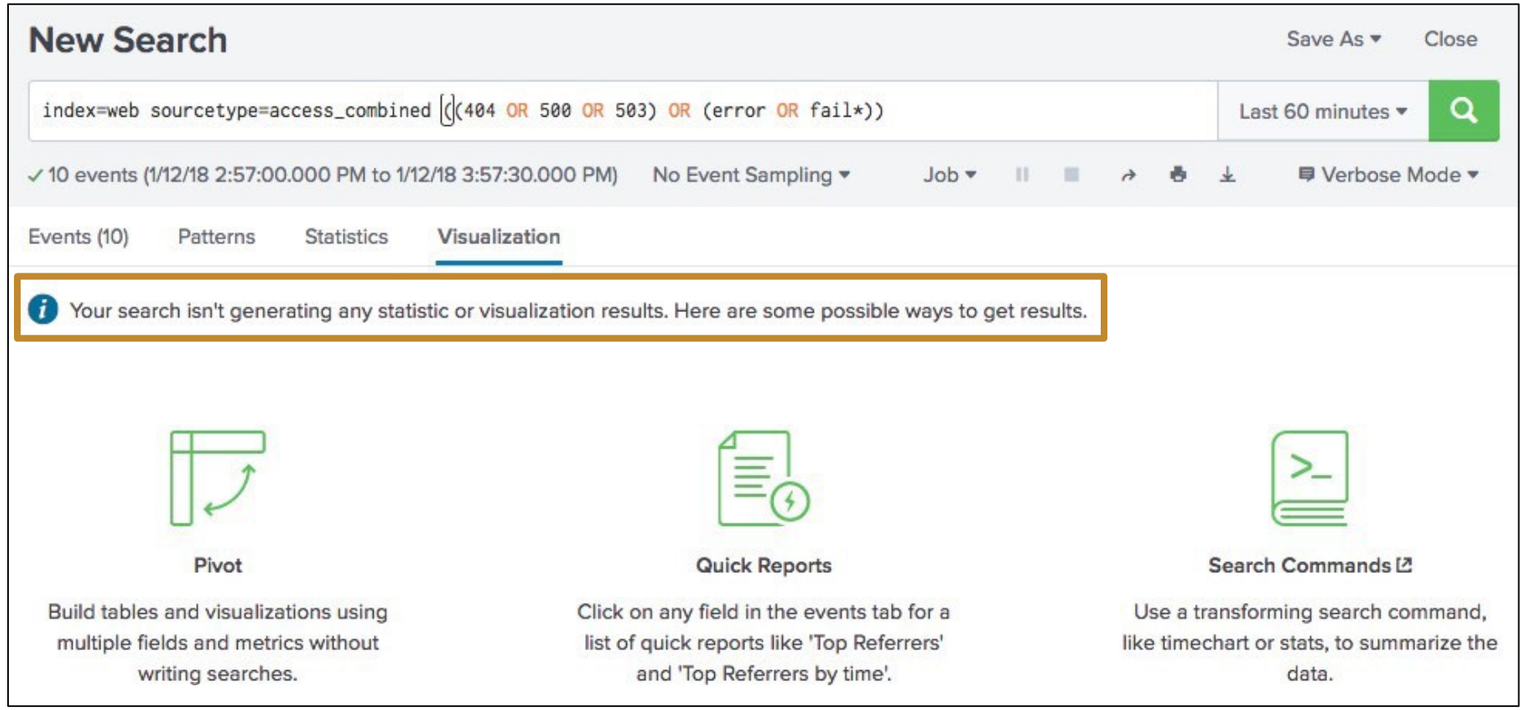
- When a search returns statistical values, results can be viewed with a wide variety of visualization types
  - Statistics table
  - Charts: Line, column, pie, etc
  - Single value, gauges
  - Maps
  - Many more





Viewing Results as a Visualization

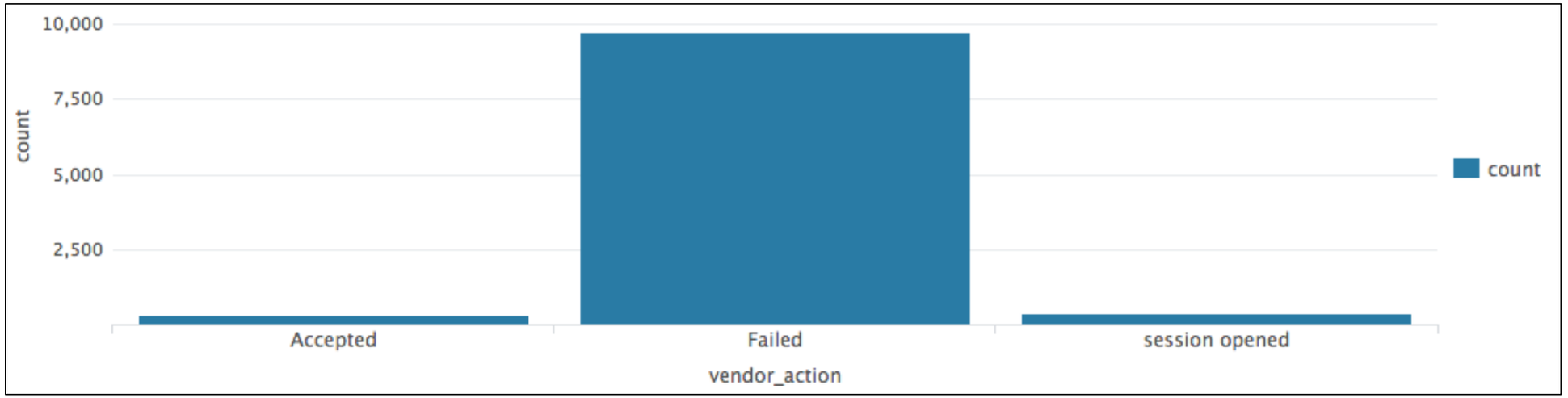
- Not all searches can be visually represented
- A data series is a sequence of related data points that are plotted in a visualization
- Data series can generate any statistical or visualization results



Data Structure Requirements – Single Series

- Most visualizations require search results structured as tables, with at least two columns, a **single series**
  - Leftmost column provides x-axis values
  - Subsequent columns provide numeric y-axis values for each series in the chart

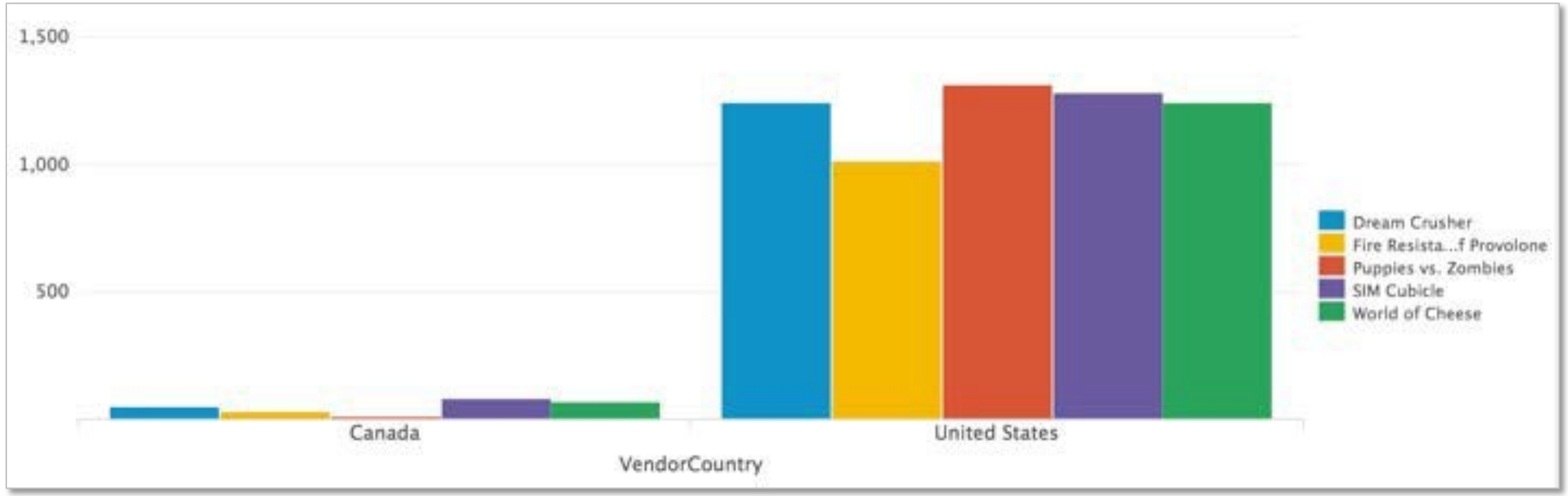
vendor_action	count
Accepted	301
Failed	9740
session opened	400



Data Structure Requirements – Multi-Series

To get **multi-series** tables, you need to set up the underlying search with reporting search commands like **chart** or **timechart**

VendorCountry	Dream Crusher	Fire Resistance Suit of Provolone	Puppies vs. Zombies	SIM Cubicle	World of Cheese
Canada	10	3	2	13	13
United States	230	202	230	228	223



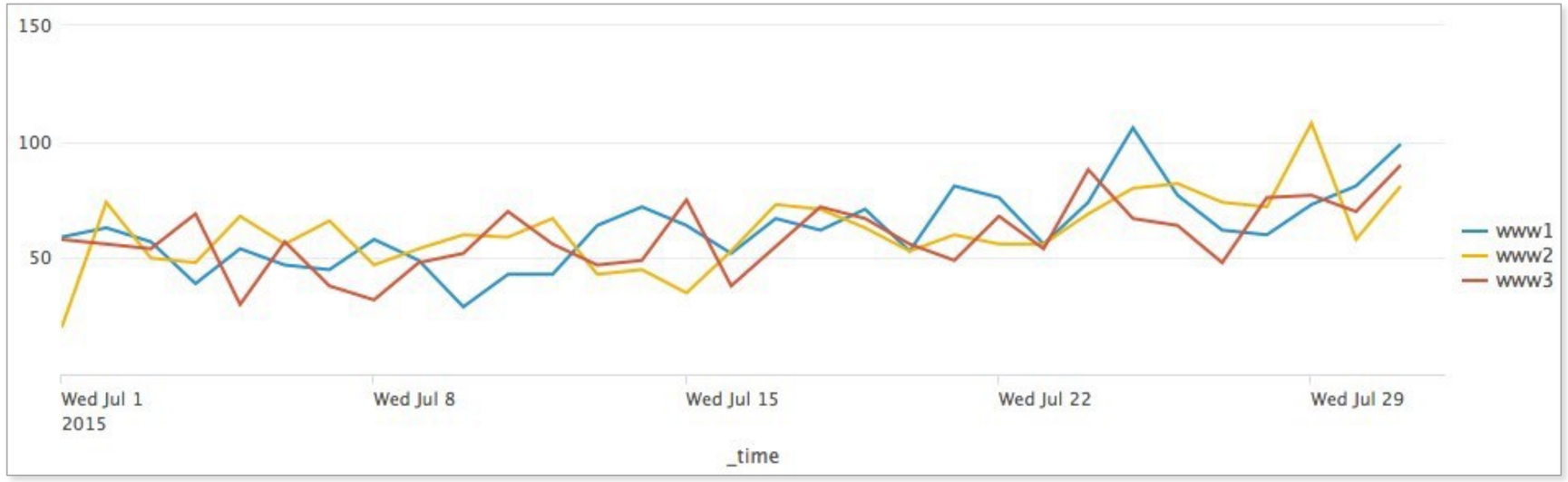

```
index=sales sourcetype=vendor_sales VendorID<4000
| chart count over VendorCountry by product_name limit=5 useother=f
```

Last 30 days

Data Structure Requirements – Time Series

- **Time series** displays statistical trends over time
- Can be single-series or multi-series

_time	www1	www2	www3
2017-12-01	168	190	158
2017-12-02	156	171	168
2017-12-03	138	150	151
2017-12-04	169	162	195
2017-12-05	135	170	150

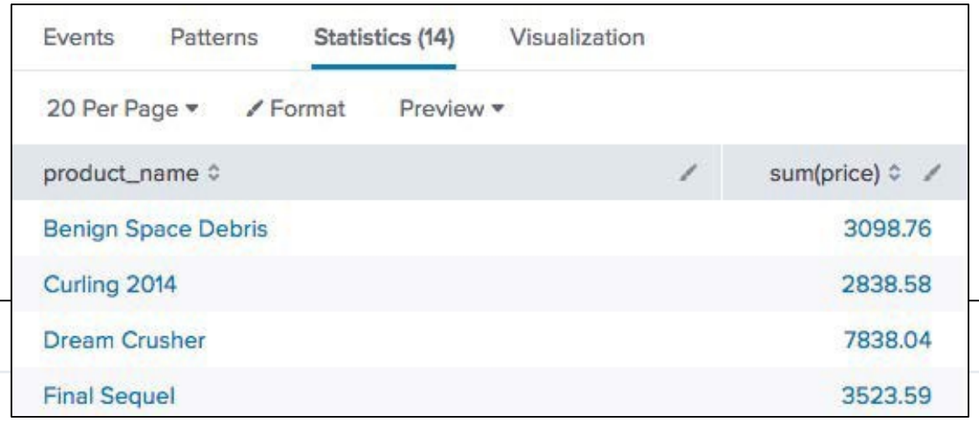



```
index=web sourcetype=access_combined action=purchase status=200
| timechart count by host
```

Previous month

Viewing Results as a Chart

- There are seven chart types:
  - Line
  - Area
  - Column
  - Bar
  - Bubble
  - Scatter
  - Pie



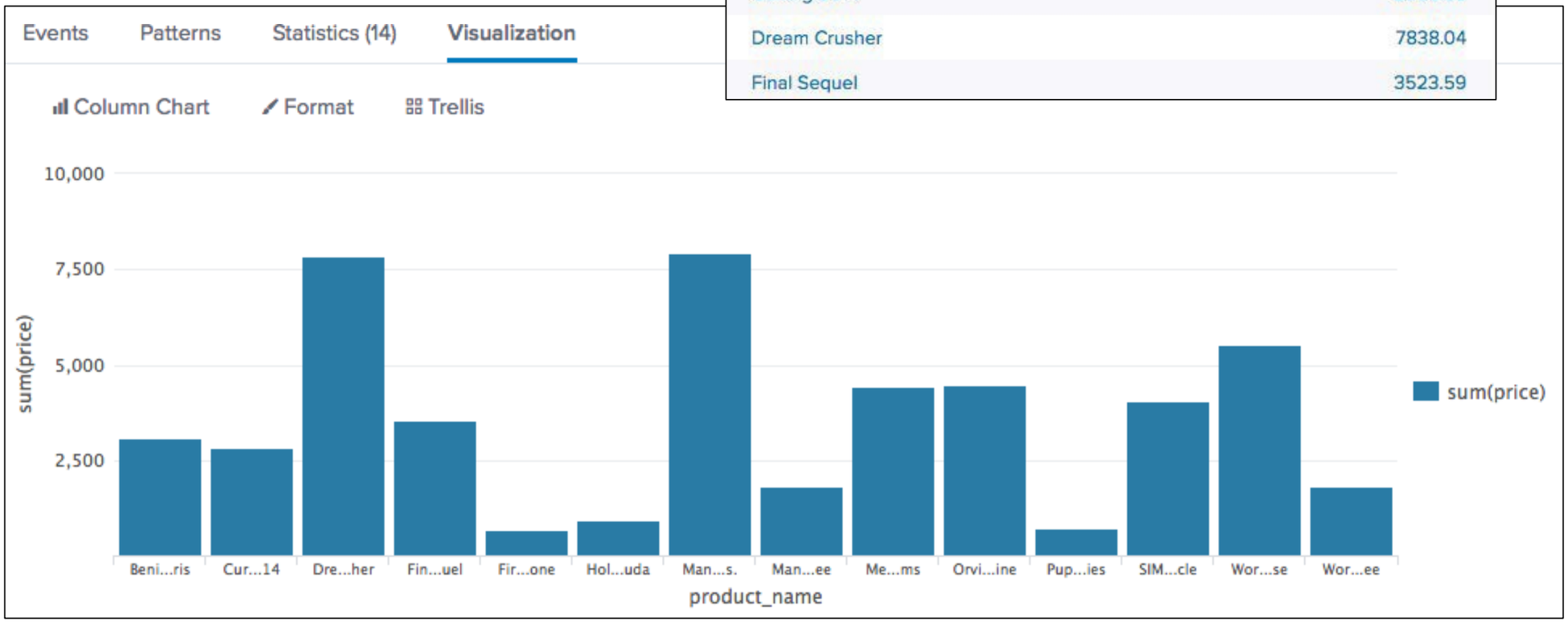


chart Command

- chart command can display any series of data that you want to plot
- You decide which field to plot on the x-axis
  - The function defines the value of the y-axis, therefore it should be numeric
  - The first field after the over clause is the x-axis
  - Using the over and by clauses divides the data into sub-groupings
    - The values from the by clause display in the legend

chart avg(bytes) over host

  - The host values display over the x-axis

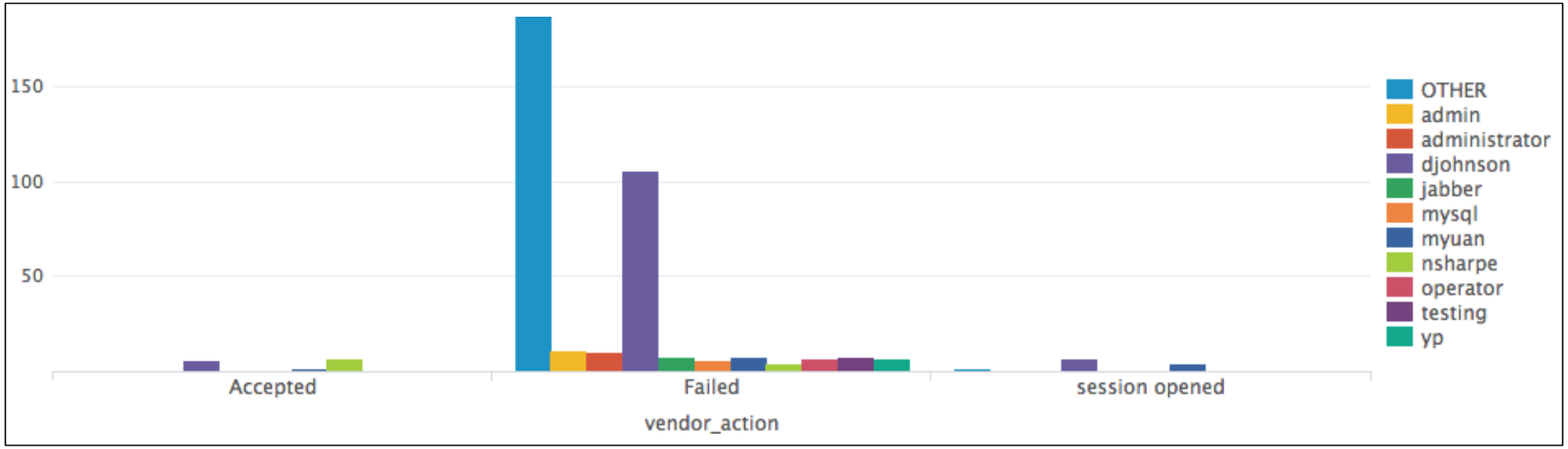
chart avg(bytes) over host by product_name

  - The host field is the x-axis and the series is further split by product_name

chart Command – over *field* by *field*

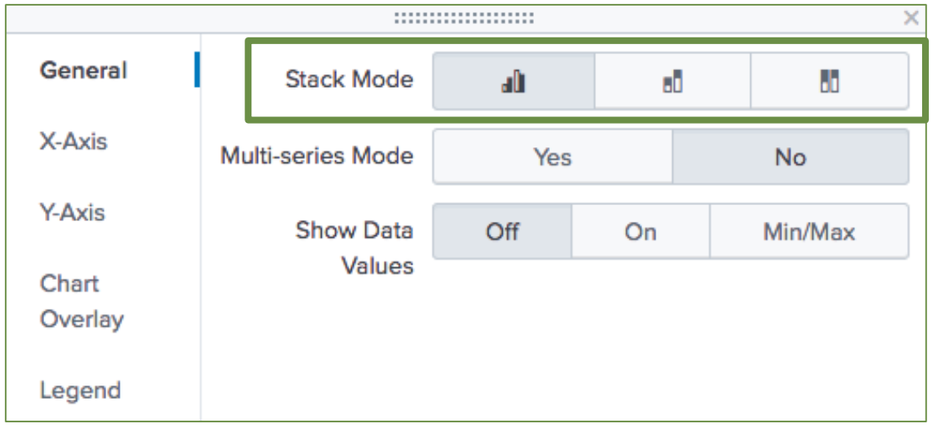
- You can use the **by** clause with the **over** clause to split results (over vendor_action by user)
- Alternatively, you can just use two **by** clauses (by vendor_action, user)
- You can only split chart results over TWO dimensions (unlike stats results)

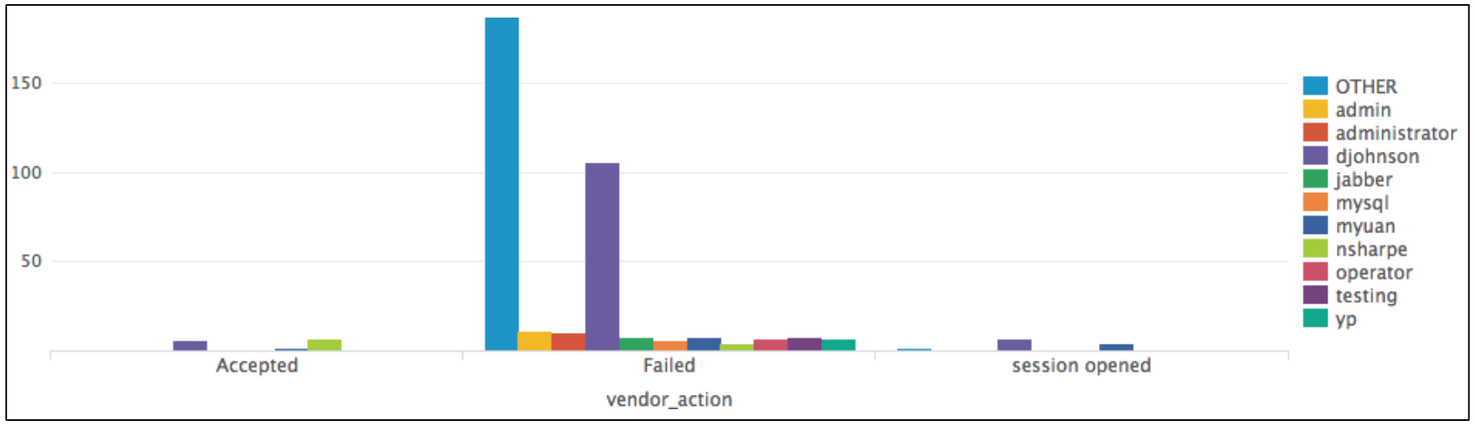
```
index=security sourcetype=linux_secure
| chart count over vendor_action by user
```



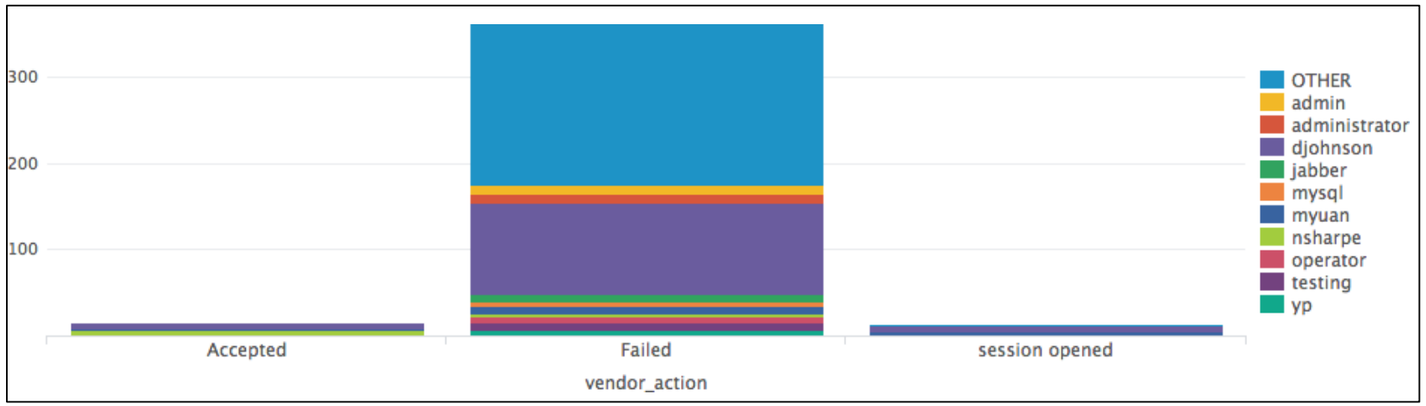
Stack Mode

Stack Mode **OFF**



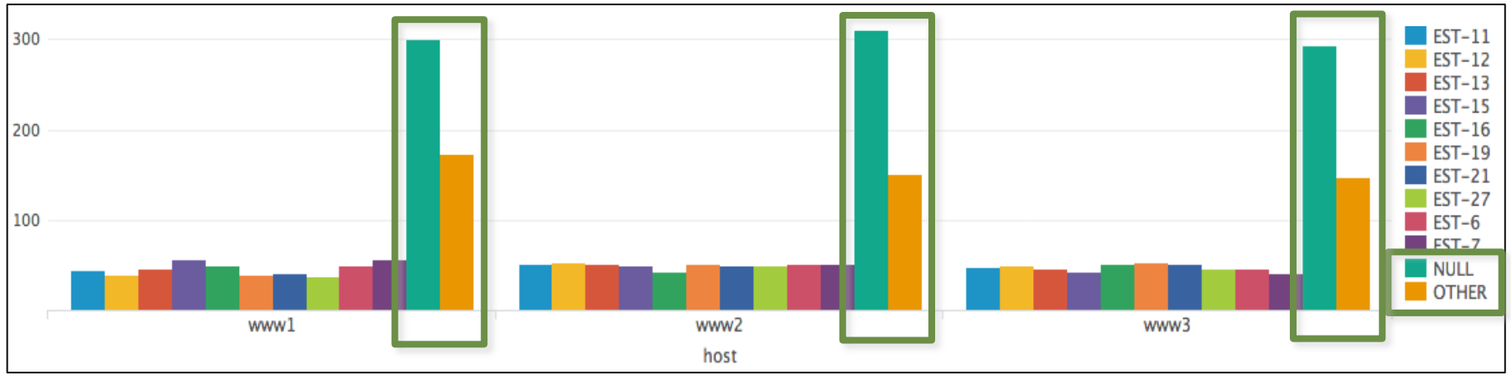


Stack Mode **ON**

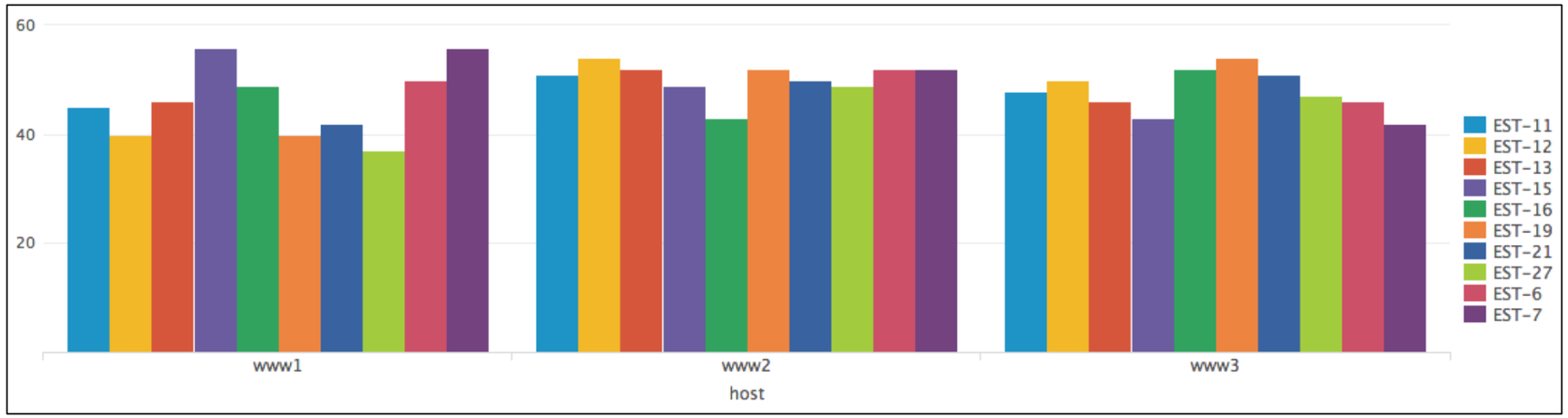


Omitting NULL and OTHER Values

- To remove empty (NULL) and OTHER field values from the displa use these options:
  - useother=f
  - usenull=f



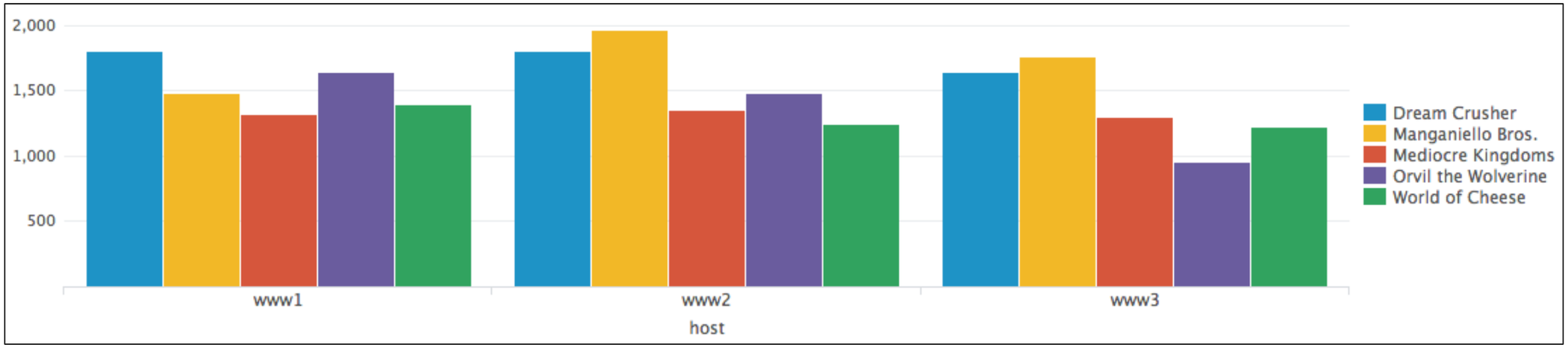

```
index=web sourcetype=access_combined status>399
| chart count over host by itemId
  useother=f usenull=f
```



Limiting the Number of Values

- To adjust the number of plotted series, use the `limit` argument
- For unlimited values, use `limit=0`

```
index=web sourcetype=access_combined
  action=purchase status=200
| chart sum(price) over host
  by product_name limit=5 useother=f
```

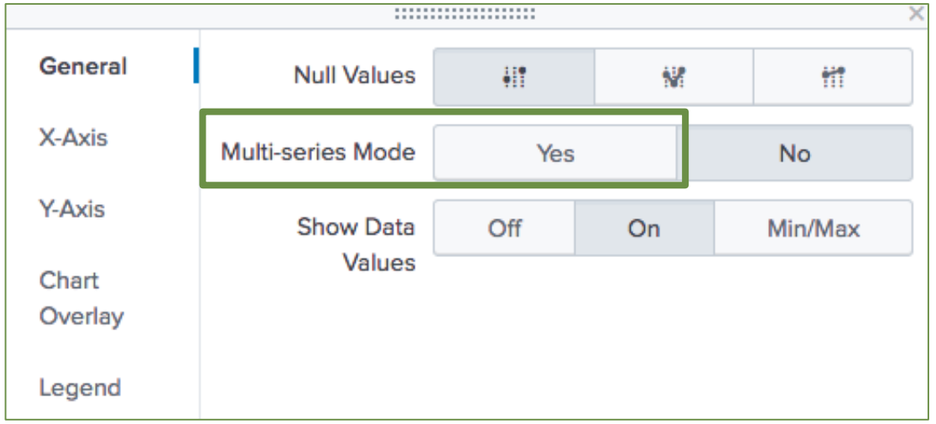


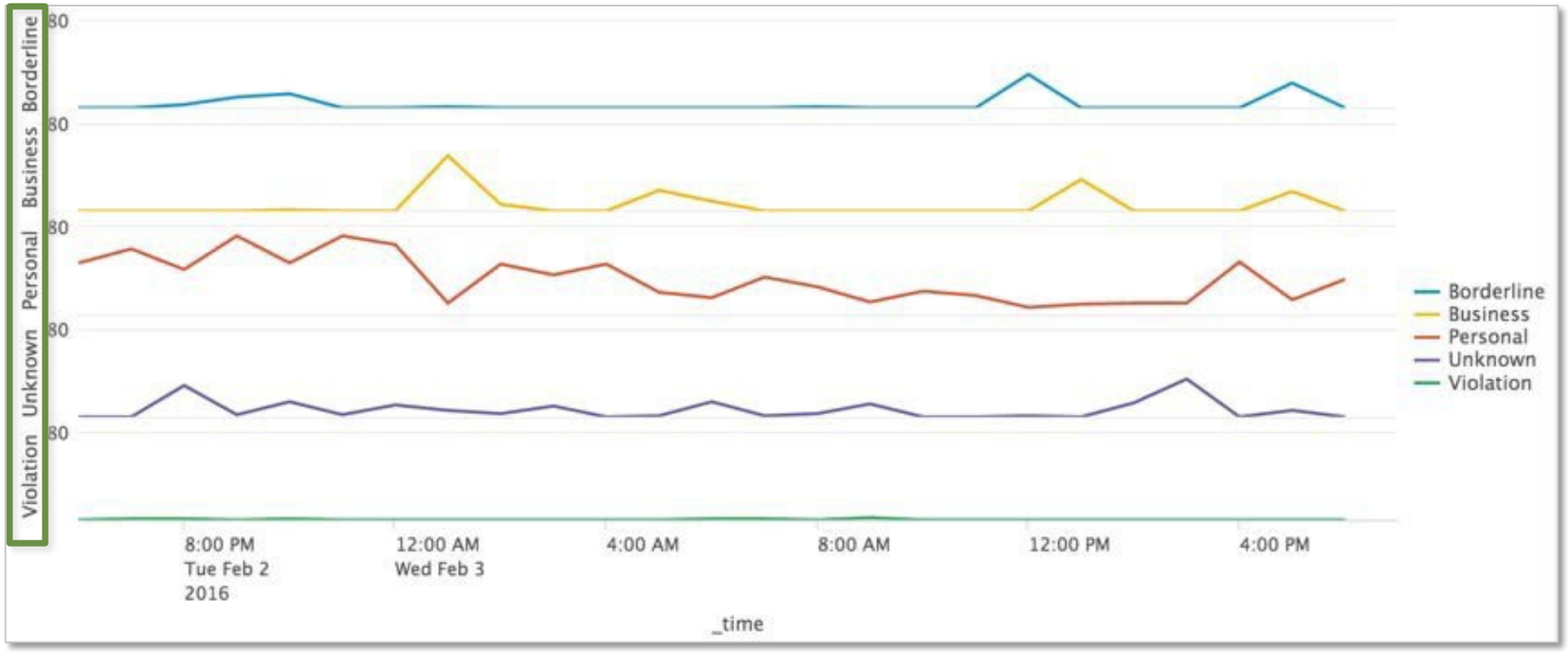
timechart Command – Overview

- timechart command performs statistical aggregations against time
- Plots and trends data over time
- _time is always the x-axis
- You can optionally split data using the by clause for one other field
  - Each distinct value of the split by field is a separate series in the chart
- Timecharts are best represented as line or area charts

timechart Command – Multi-series: Yes

- Setting multi-series mode to **Yes** causes the y-axis to split for each field value
- y-axis is divided into sections, each spanning the same max and min count

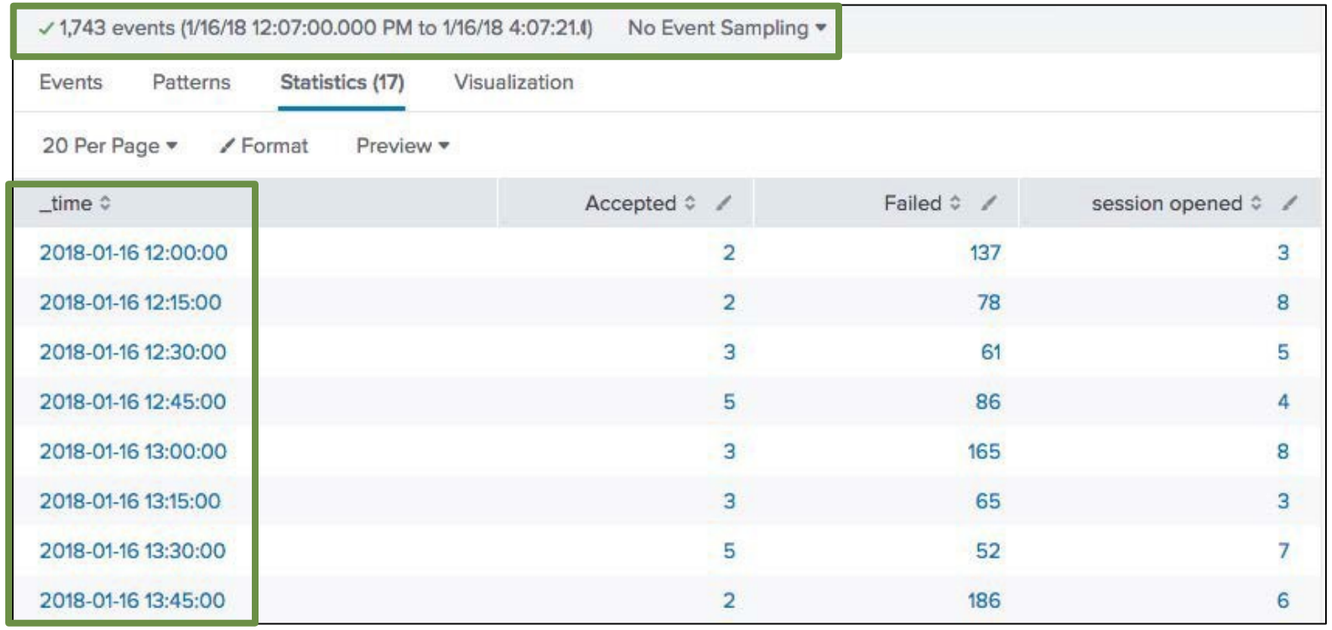




timechart Command – Adjusting the Sampling Interval

- The timechart command "buckets" the values of the _time field
  - This provides dynamic sampling intervals, based upon the time range of the search
- Example defaults:
  - Last 60 minutes uses span=1m
  - Last 24 hours uses span=30m
- Adjust the interval using the span argument, e.g. span=15m

```
index=security sourcetype=linux_secure
vendor_action=*
| timechart span=15m count by vendor_action
```

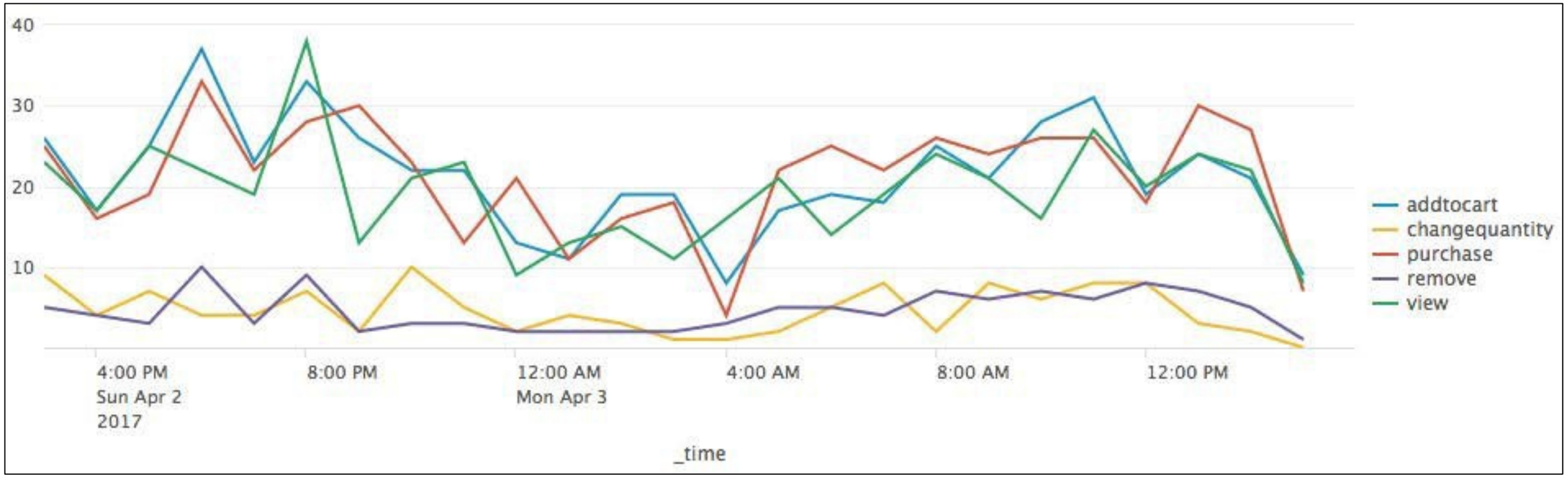


_time	Accepted	Failed	session opened
2018-01-16 12:00:00	2	137	3
2018-01-16 12:15:00	2	78	8
2018-01-16 12:30:00	3	61	5
2018-01-16 12:45:00	5	86	4
2018-01-16 13:00:00	3	165	8
2018-01-16 13:15:00	3	65	3
2018-01-16 13:30:00	5	52	7
2018-01-16 13:45:00	2	186	6

timechart Command – Statistical Functions

As with the stats and chart commands, you can apply statistical functions to the timechart command

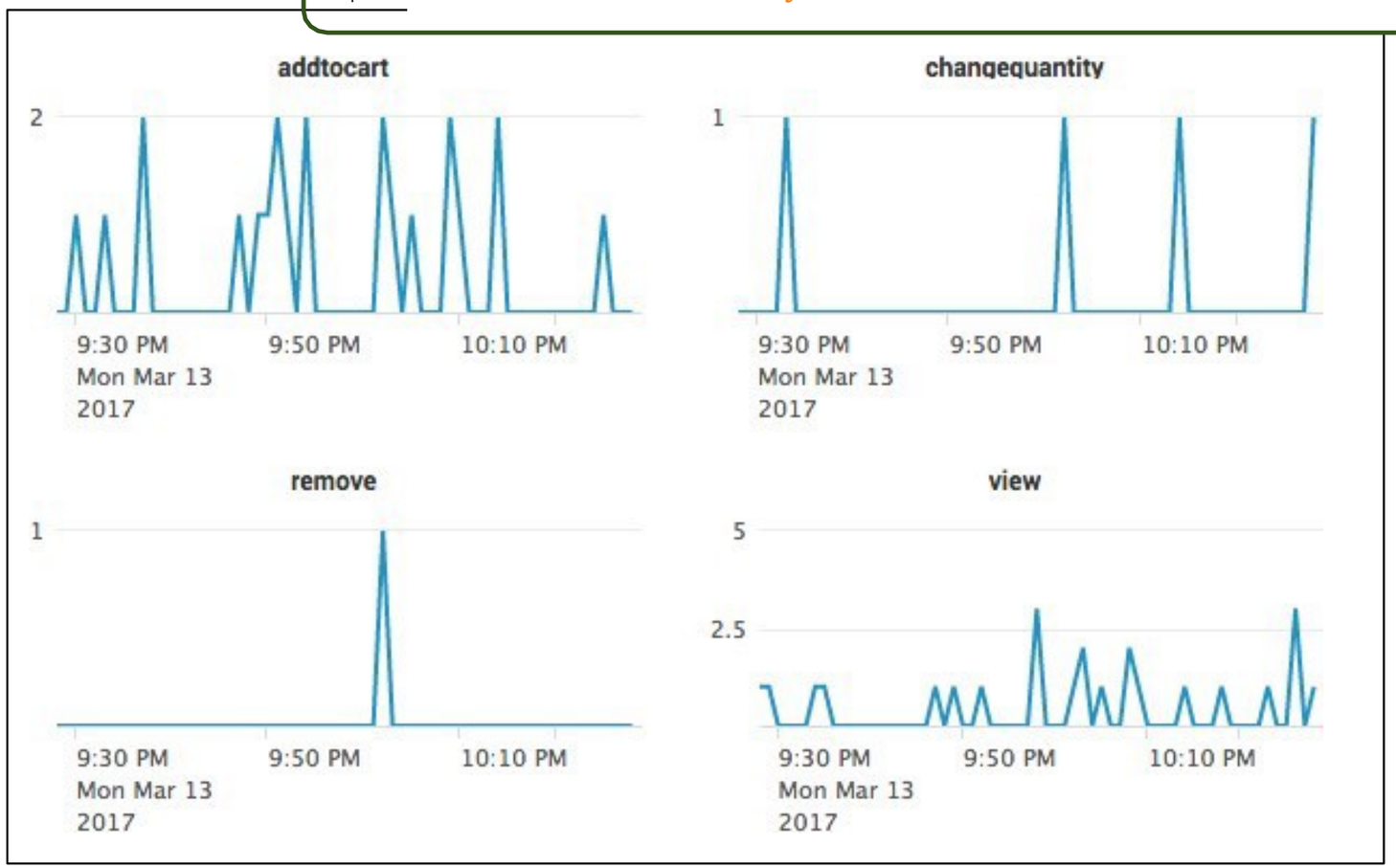
```
index=web sourcetype=access_combined action=*
| timechart span=1h count by action
```

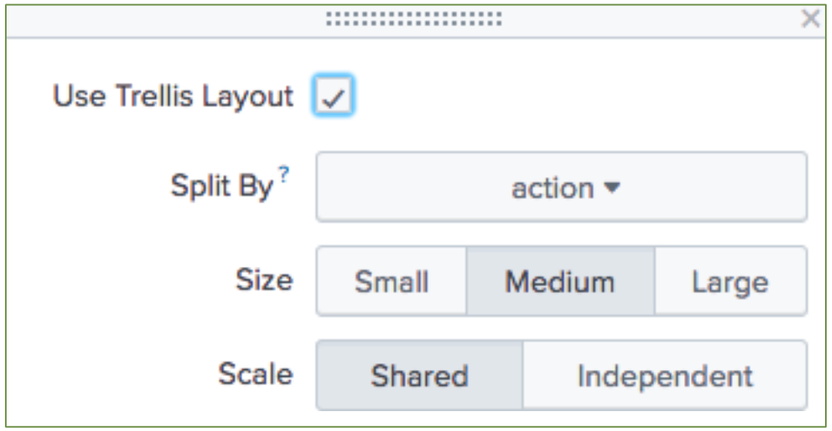


Trellis Layout

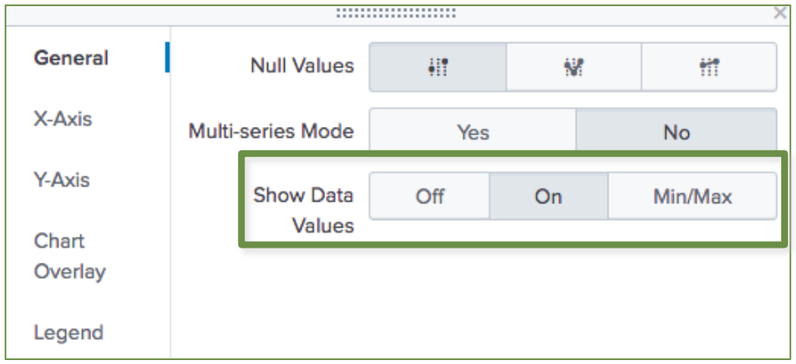
- Display multiple charts based on one result set
- Allows visual comparison between different categories
- Data only fetched once

```
index=web sourcetype=access_combined action=*
| timechart count by action
```

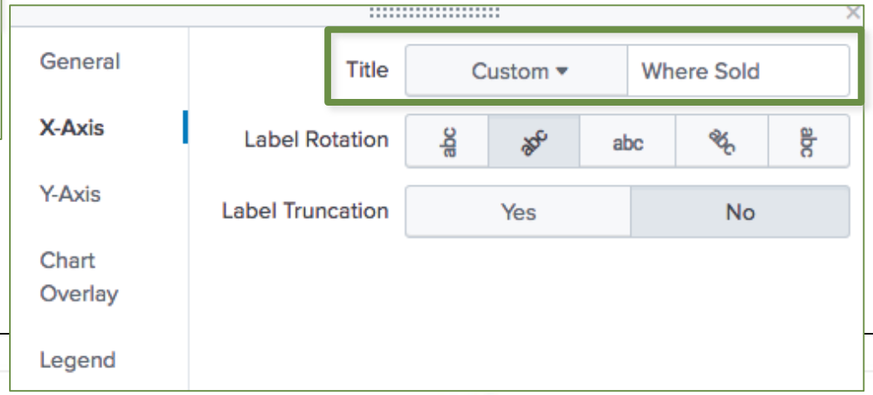


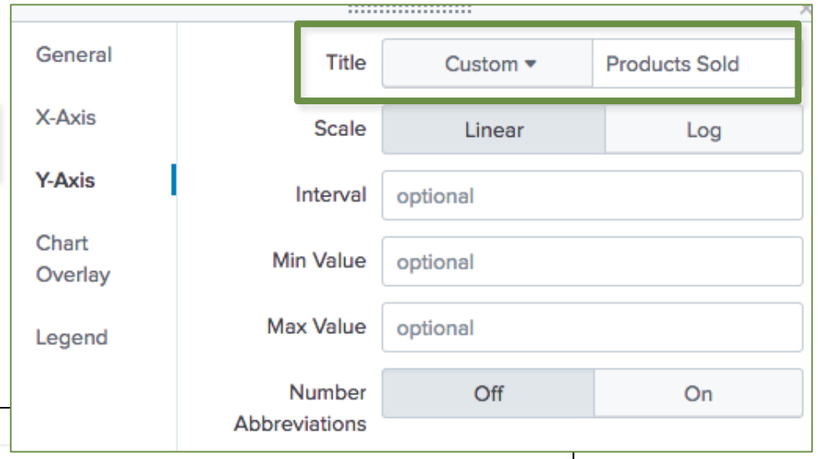


Visualization Formatting



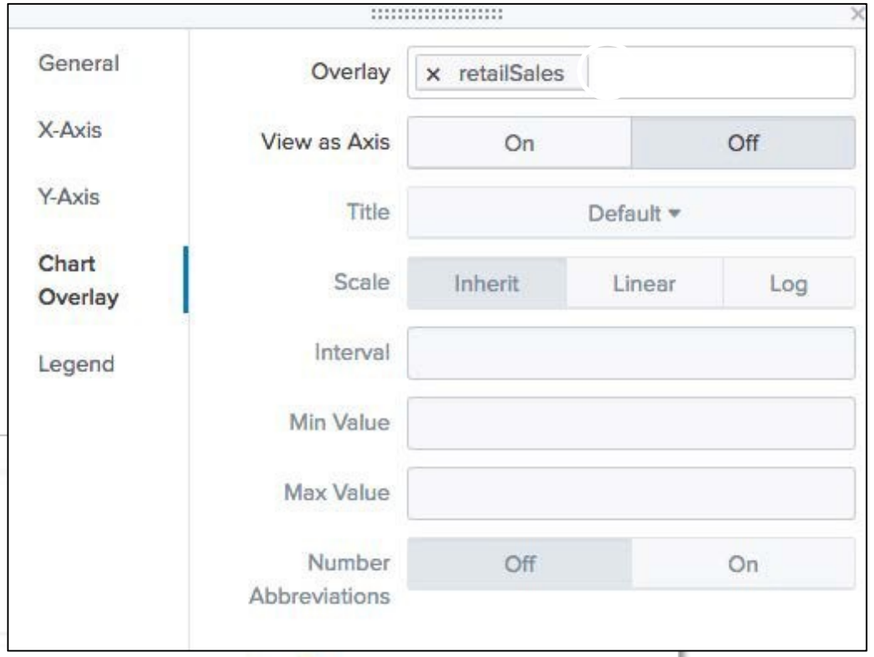

```
index=web sourcetype=access_combined
| chart count by host
```

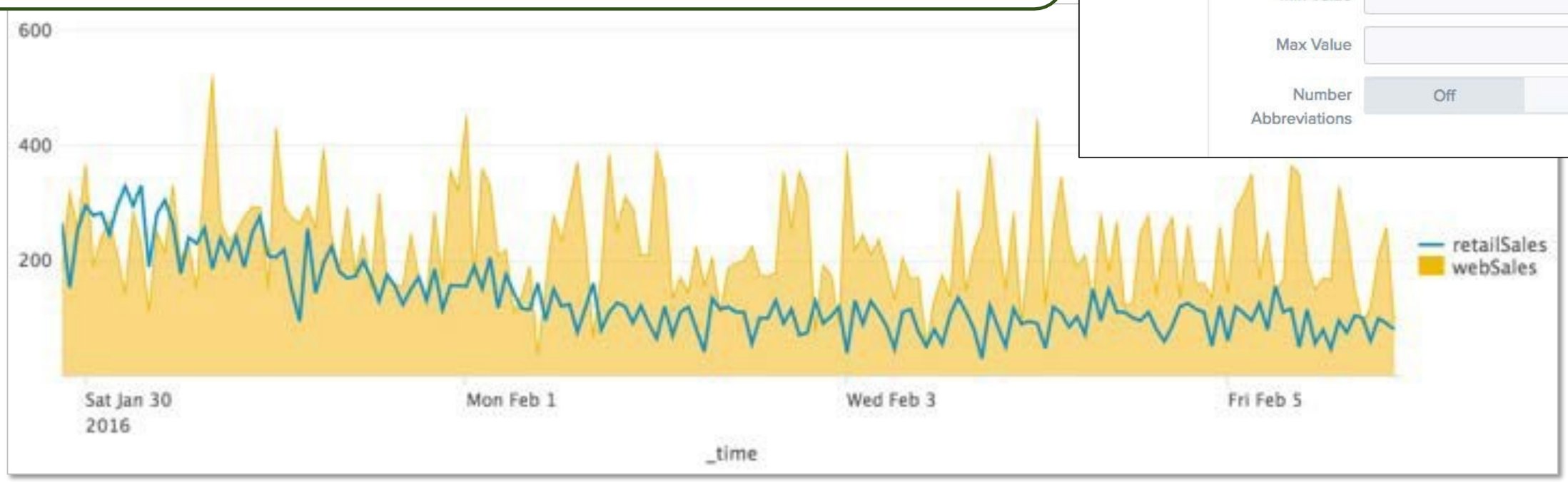




Formatting – Chart Overlay

```
(index=web sourcetype=access_combined action=purchase status<400)
OR (index=sales sourcetype=vendor_sales)
| timechart span=1h sum(price) by sourcetype
| rename access_combined as webSales, vendor_sales as retailSales
```





Transforming Command Summary

Feature	stats	chart	timechart
Multi-level breakdown [by clause]	Many	2	1
Limit # series shown	NA	limit=*n* *Default=10*	limit=*n* *Default=10*
Filter other series	NA	useother=f	useother=f
Filter null values	NA	usenull=f	usenull=f
Set time value on x axis	NA	NA	span

Transforming Command Summary (cont.)

To count the frequency of a field(s), use `top/rare`

```
index=security
sourcetype=linux_secure
 | top src_ip, user, vendor_action
```

src_ip	user	vendor_action	app	count	percent
235.166.61.39	nsharpe	Failed	sshd	122	7.850708
44.248.239.252	myuan	Failed	sshd	113	7.271557
16.201.248.137	nsharpe	Failed	sshd	113	7.271557
43.189.188.26	myuan	Failed	sshd	107	6.885457

Transforming Command Summary (cont.)

Use `stats` to calculate statistics for two or more `by` fields (non time- based)

```
index=security sourcetype=linux_secure
| stats count by src_ip, user, vendor_action, app
```

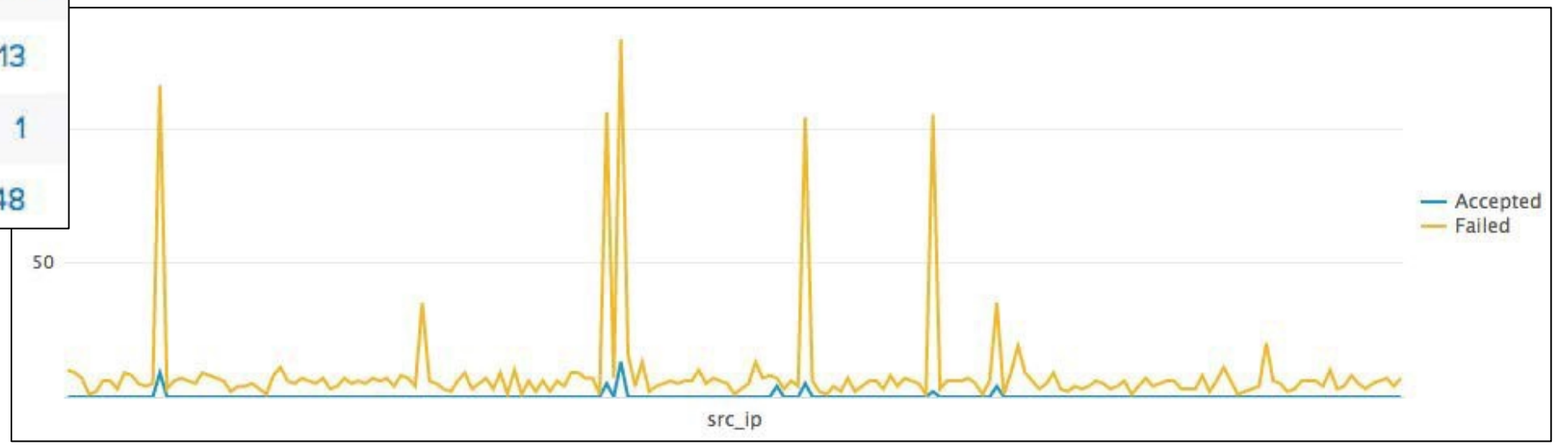
src_ip	user	vendor_action	app	count
105.131.160.148	djohnson	Failed	sshd	100
107.3.146.207	admin	Failed	sshd	2
107.3.146.207	administrator	Failed	sshd	3
107.3.146.207	angel	Failed	sshd	1
107.3.146.207	bin	Failed	sshd	1
107.3.146.207	couchdb	Failed	sshd	1
107.3.146.207	customer	Failed	sshd	1
107.3.146.207	db	Failed	sshd	1
107.3.146.207	desktop	Failed	sshd	4
107.3.146.207	divine	Failed	sshd	1

Transforming Command Summary (cont.)

- To calculate statistics with an arbitrary field as the x-axis (not `_time`), use `chart`
  - When you use a `by` field, the output is a table
  - Each column represents a distinct value of the split-by field

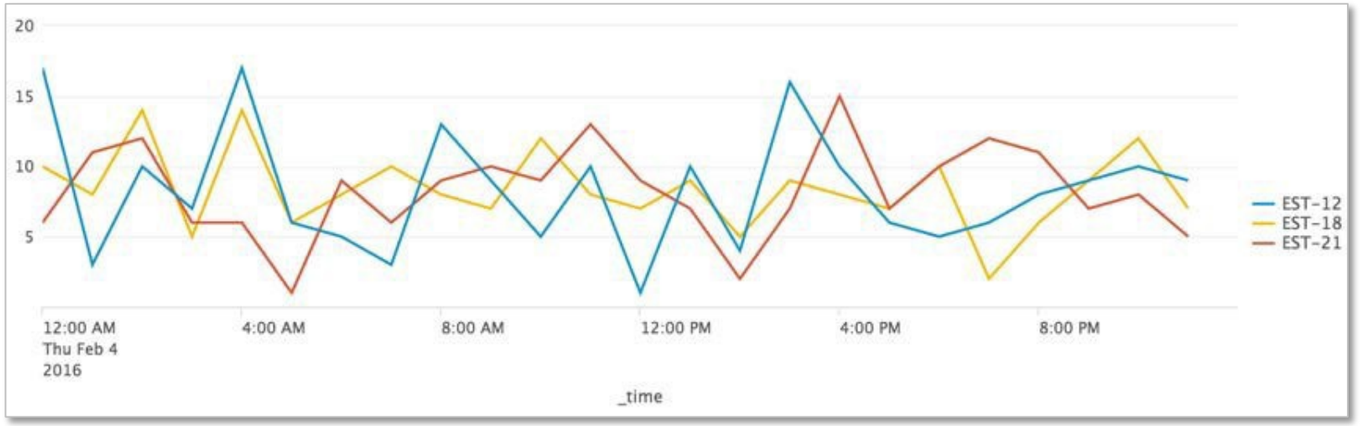
```
index=security sourcetype=linux_secure
| chart count over src_ip
  by vendor_action
```

src_ip	Accepted	Failed
15.170.91.198	5	109
16.201.248.137	3	113
17.186.26.16	0	1
17.186.26.167	41	148



Transforming Command Summary (cont.)

- Use `timechart` to calculate statistics with `_time` as the x-axis
- If a `by` field is used, the output is a table
- Each column represents a distinct value of the split-by field




```
... | timechart span=1h count by itemId limit=3 useother=f
```

_time	EST-13	EST-15	EST-7	NULL
2018-01-16 12:00	7	5	4	28
2018-01-16 13:00	15	6	7	53
2018-01-16 14:00	8	20	15	42
2018-01-16 15:00	3	8	9	38
2018-01-16 16:00	5	8	4	34

More Visualizations

trendline Command

- Allows you to overlay a computed moving average on a chart
- trendline computes the moving averages of a field

  trendline *<trendtype><period>(field)* [AS *newfield*]

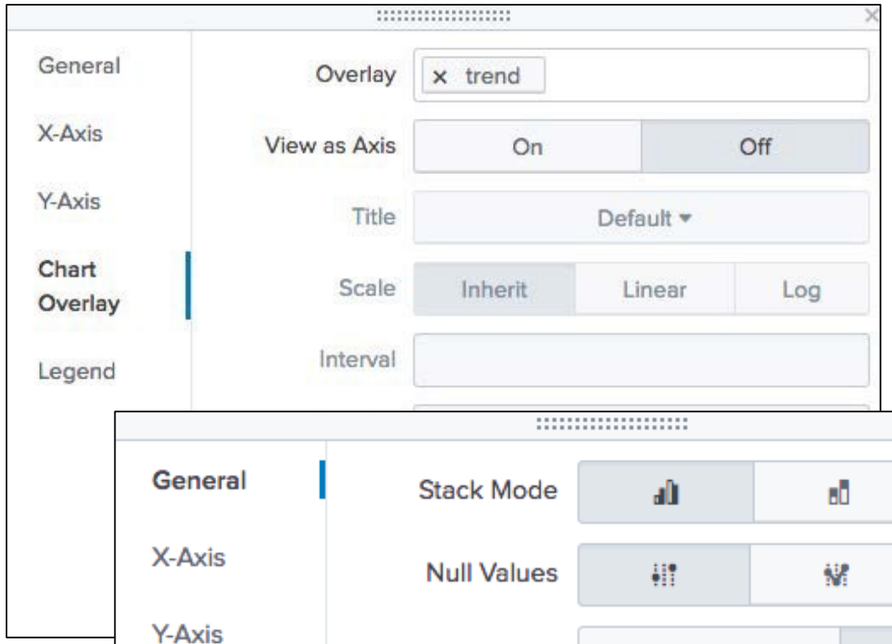
- *trendtype*:
  - sma - simple moving average
  - ema - exponential moving average
  - wma - weighted moving average

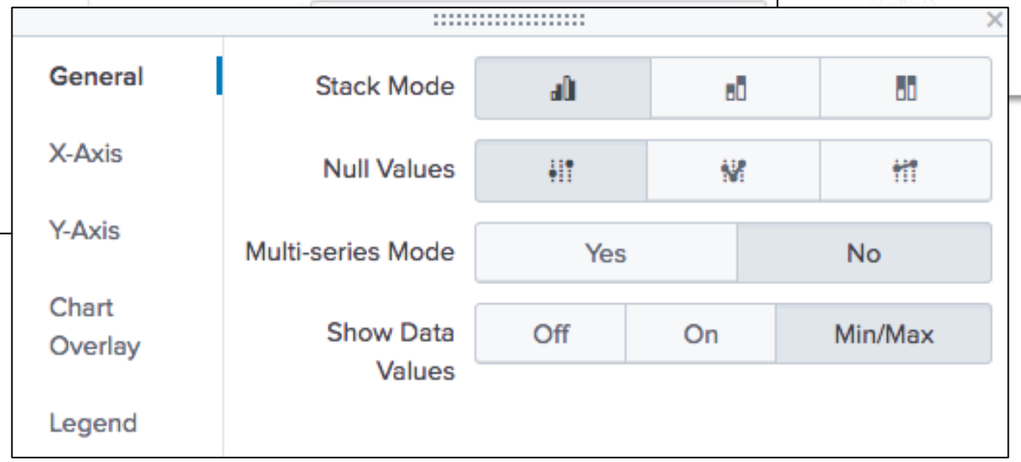
trendline Command (cont.)

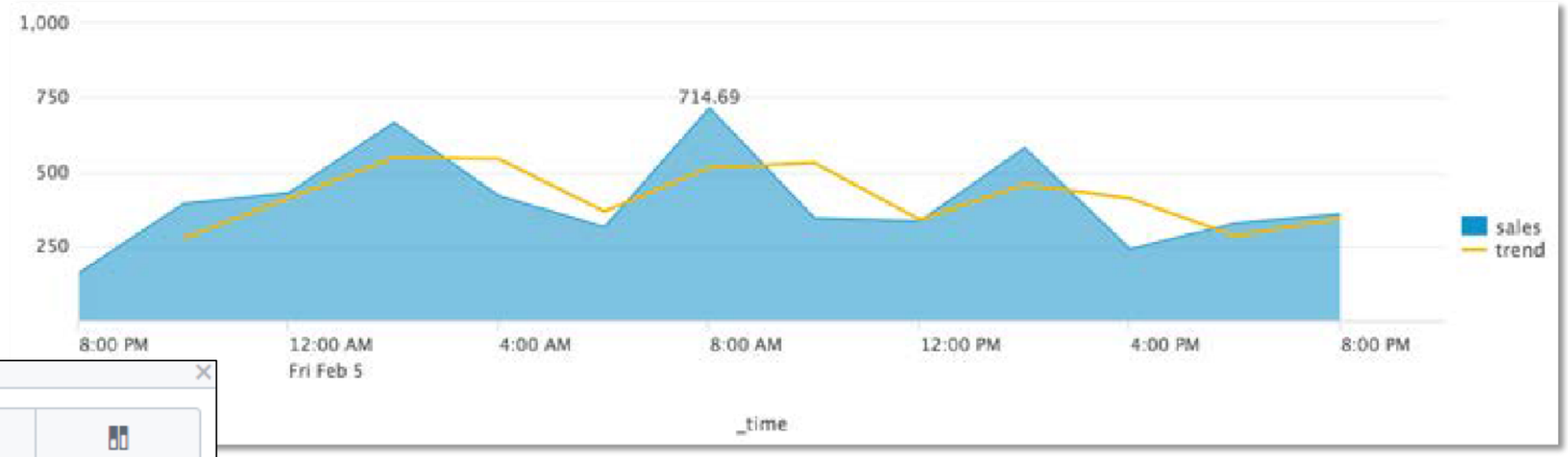
- Must define the *period* over which to compute the trend
- *period* must be an integer between 2 and 10000
  - For example, sma2(sales) is valid
  - But sma(sales) would *fail* as it is missing an integer, the defining period

trendline Command – Example

```
index=web sourcetype=access_combined action=purchase status=200
| timechart span=2h sum(price) as sales
| trendline sma2(sales) as trend
```





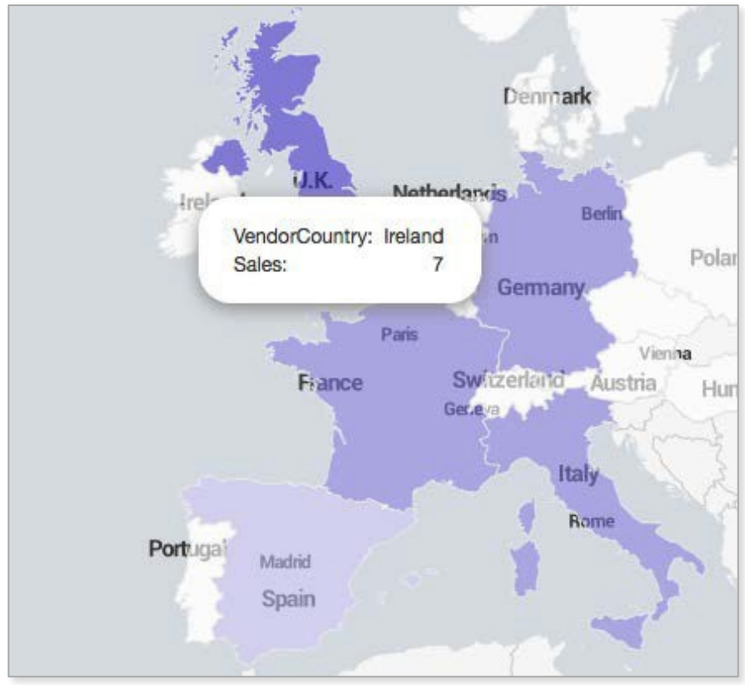


Viewing Results as a Map

There are two map types

Cluster Map

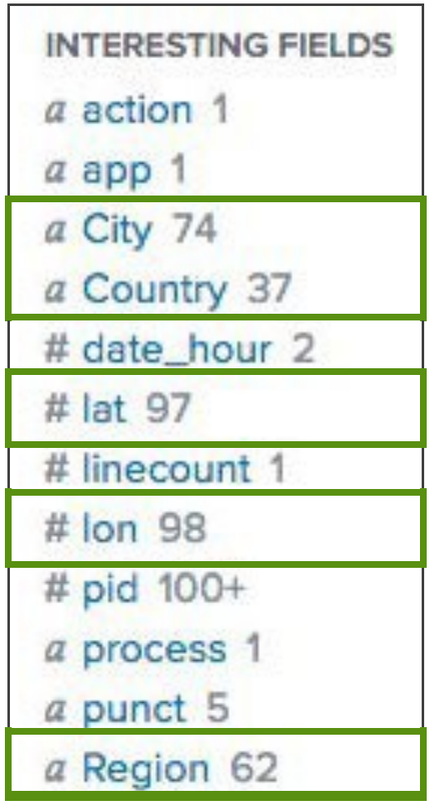
Choropleth Map



iplocation Command

```
index=security sourcetype=linux_secure (fail* OR invalid)
| iplocation src_ip
```

- Use iplocation to look up and add location information to an event
  - This information includes city, country, region, latitude and longitude
- Not all of the information is available for all ip address ranges
- Automatically defines the default lat and lon fields required by geostats



geostats Command

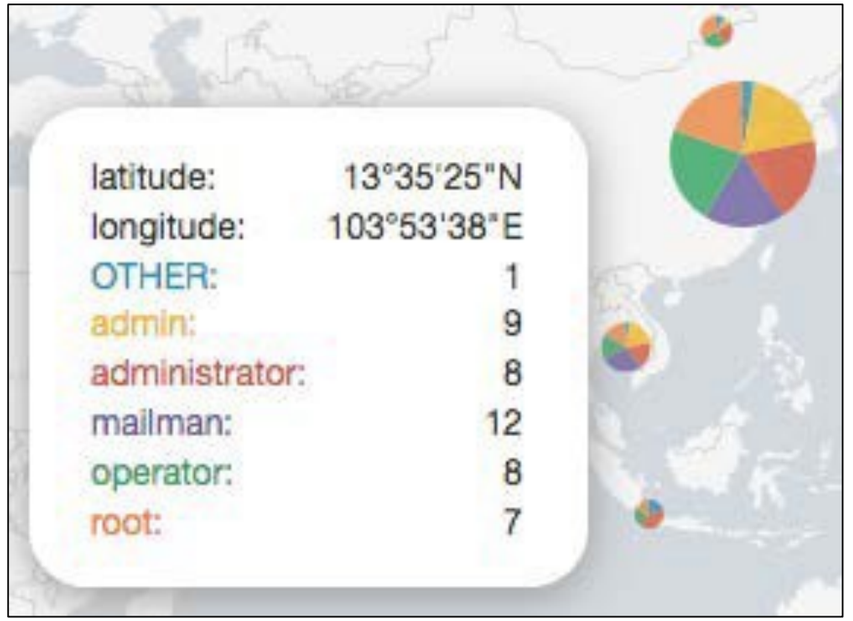
- Use geostats to compute statistical functions and render a cluster map

  geostats [latfield=*string*] [longfield=*string*] [*stats-agg- term*]* [by-clause]

- Data must include latitude and longitude values
- Define the *latfield* and *longfield* only if they differ from the default lat and lon fields
- To control the column count:
  - On a global level, use the globallimit argument
  - On a local level, depending on where your focus is (i.e., where you've zoomed in), use the locallimit argument

geostats Command – Example

```
index=security sourcetype=linux_secure
(fail* OR invalid)
| iplocation src_ip
| geostats globallimit=5 count by user
```



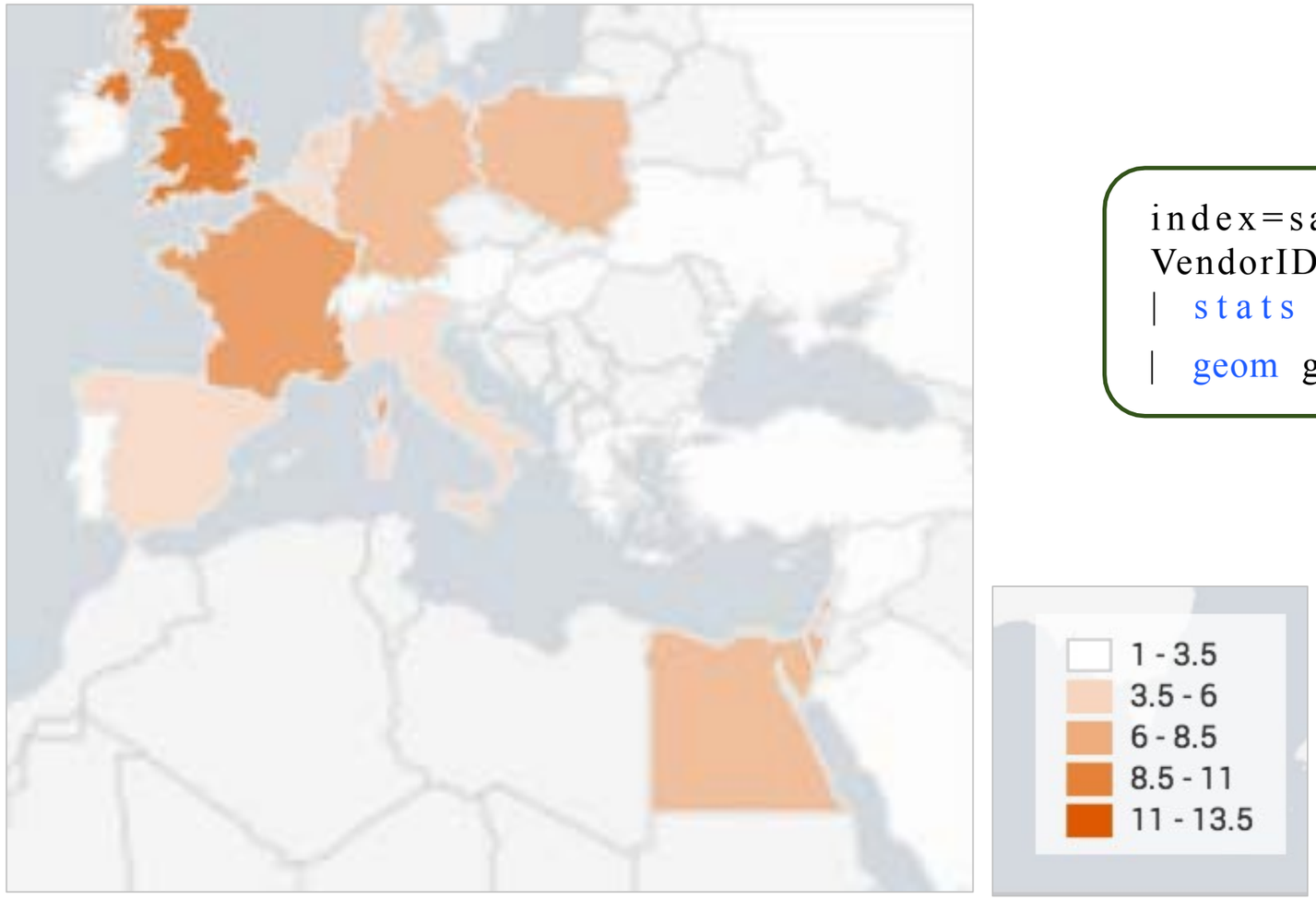
geobin	latitude	longitude	OTHER	admin	administrator	mailman	operator	root
bin_id_zl_0_y_2_x_2	-25.96763	-53.70309	3	4	5	4	2	
bin_id_zl_0_y_2_x_4	-29.00000	24.00000	1		1	3	3	1
bin_id_zl_0_y_2_x_7	-33.22925	151.62210	1	3	2		1	7
bin_id_zl_0_y_3_x_3	-8.05000	-34.90000	110					

Choropleth Map

- Uses shading to show relative metrics, such as sales, network intruders, etc. for predefined geographic regions
- To define regional boundaries, you must have either a:
  - KML (Keyhole Markup Language) file
  - KMZ (compressed Keyhole Markup Language) file
- Splunk ships with:
  - `geo_us_states`, United States
  - `geo_countries`, countries of the world

`...| geom [`*`featureCollection`*`] [featureIdField=`*`string`*`]`

geom Command




```
index=sales sourcetype=vendor_sales
VendorID > 4999 AND VendorID < 6000
| stats count as Sales by VendorCountry
| geom geo_countries featureIdField=VendorCountry
```

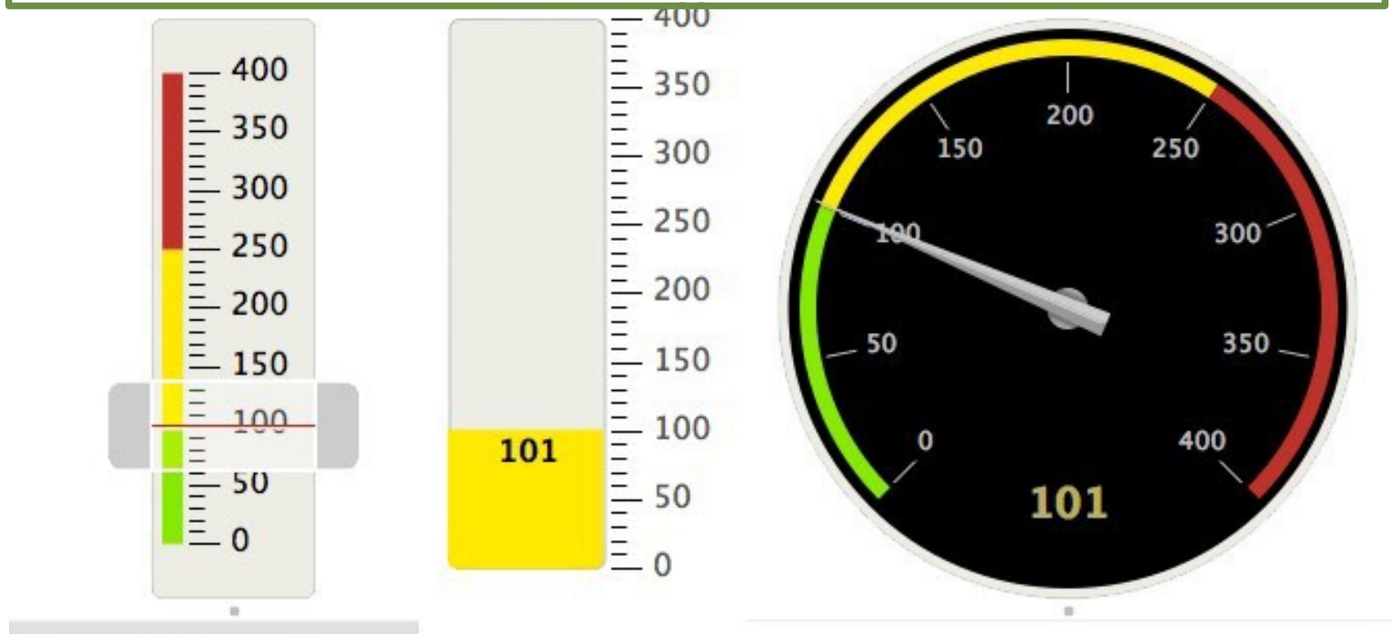
Viewing Results as a Single Value

Single value visualizations provide various formatting options

```
index=security sourcetype=linux_secure vendor_action=failed
| stats count
```

101

Marker Filler Radial



Single Value Visualizations: timechart

- With the timechart command, you can add a sparkline and a trend
- A **sparkline** is an inline chart
  - It is designed to display time-based trends associated with the primary key
- The **trend** shows the direction in which values are moving
  - It appears to the right of the single value

```
index=security sourcetype=linux_secure
fail* OR invalid
| timechart span=15m count(vendor_action)
```

Adding Totals Using Format Options

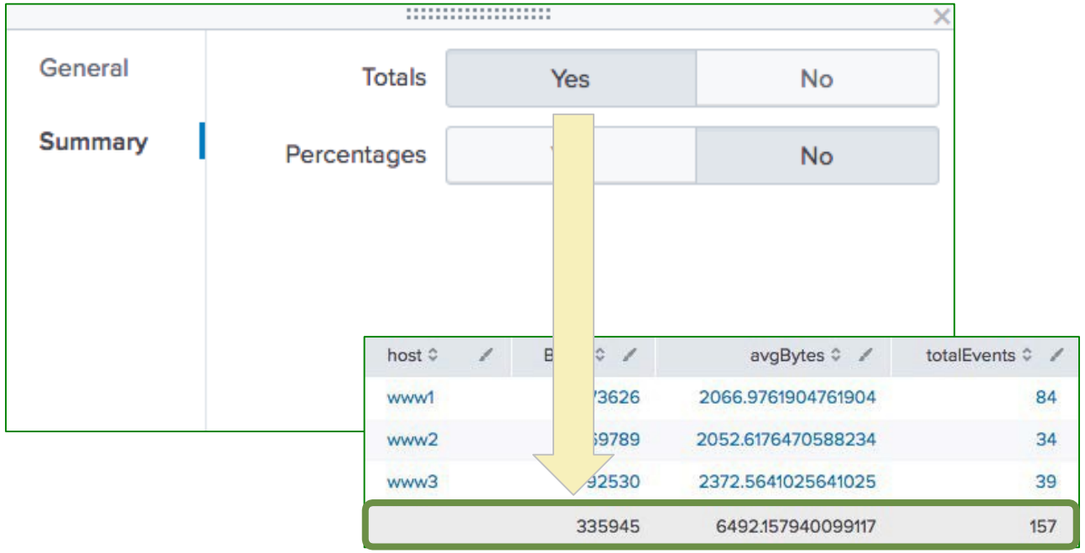
- Automatically total every column using the Format options
- When using this approach, you:
  - Cannot indicate which column to total; all columns are always totaled
  - Cannot add labels

```
index=web sourcetype=access_combined
| stats sum(bytes) as Bytes,
  avg(bytes) as avgBytes,
  count as totalEvents by host
```

Events | Patterns | Statistics (3) | Visualization

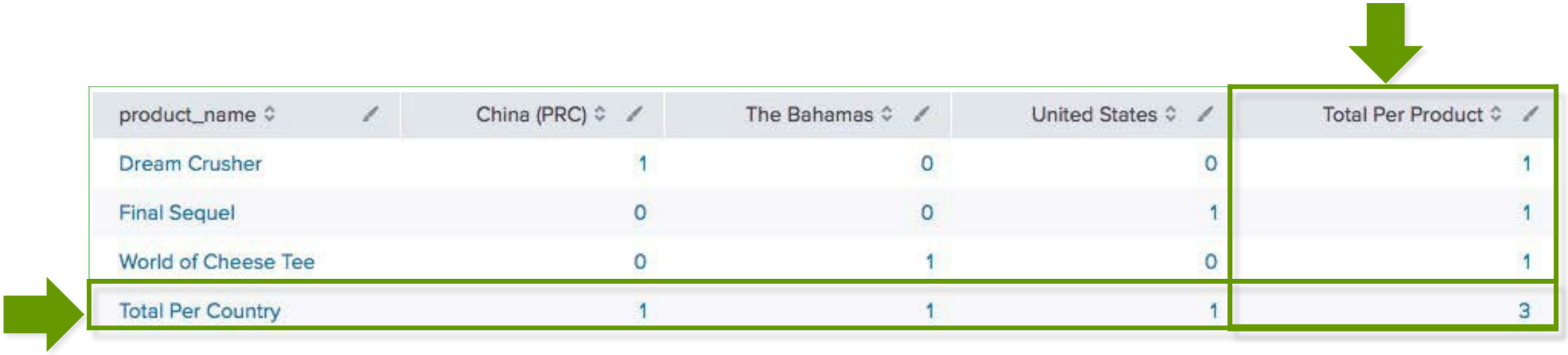
20 Per Page | Format | Preview

host	Bytes	avgBytes	totalEvents
www1	173626	2066.9761904761904	84
www2	69789	2052.6176470588234	34
www3	92530	2372.5641025641025	39



Adding Totals Using `addtotals` Command

- Alternatively, use the `addtotals` command to:
  - Compute the sum of all **or selected** numeric fields for each column and place the total in the last row
  - Compute the sum of all **or selected** numeric fields for each **row** and place the total in the last column



product_name	China (PRC)	The Bahamas	United States	Total Per Product
Dream Crusher	1	0	0	1
Final Sequel	0	0	1	1
World of Cheese Tee	0	1	0	1
Total Per Country	1	1	1	3

addtotals Command: Syntax

addtotals [row=*bool*] [fieldname=*field*] [col=*bool*][labelfield=*field*] [label=*string*] *field-list*

		Column Options	
row=true/false (Default=true)	A column is created that contains numeric totals for each row.	col=true/false (Default=false)	A row is created that contains numeric totals for each column.
fieldname=*field* (Default=Total)	Defines a string used to create a field name for the totals column.	label=*string* (Default=Total)	Defines a string used to name the totals row.
		labelfield=*fieldname*	Defines where the label string is placed. (Generally, you should make this the first

General Options	
field-list=one or more numeric fields. (Default: all numeric fields)	Defines the numeric fields to be totaled.

addtotals Command

- Do not total rows
- Total columns
- Add the label totalBytes
- Place the label under the host column
- Only total the Bytes column

```
index=web sourcetype=access_combined
| stats sum(bytes) as Bytes, avg(bytes) as
  avgBytes, count as totalEvents by host

| addtotals row=f col=t label=totalBytes
  labelfield=host Bytes
```

host	Bytes	avgBytes	totalEvents
www1	141882	2149.7272727272725	66
www2	93781	2084.0222222222224	45
www3	122834	2233.3454545454547	55
totalBytes	358497		

Filtering and Formatting Data

eval Command – Overview

- eval allows you to calculate and manipulate field values in your report

  eval *fieldname1 = expression1 [, fieldname2 = expression2...]*

- Supports a variety of functions
- Results of eval written to either new or existing field you specify
  - If the destination field exists, the values of the field are replaced by the results of eval
  - Indexed data is not modified, and no new data is written into the index
  - Field values are treated in a case-sensitive manner

eval Command

- The eval command allows you to:
  - Calculate expressions
  - Place the results in a field
  - Use that field in searches or other expressions

eval Learn More

Calculates an expression and puts the resulting value into a field. You can specify to calculate more than one expression.

Example:
... | eval velocity=distance/time

- Type

Type	Operators
Arithmetic	+ - * / %
Concatenation	+ .
Boolean	AND OR NOT XOR
Comparison	< > <= >= != = == LIKE

eval Command – Convert Values (cont.)

- Results of eval must be set to a new or existing field
- In this example:

  Calculate the number of bytes for each usage type

  Create a new field named bandwidth

  Convert the values of the Bytes field into MB by dividing Bytes field values by (1024*1024)

```
index=network sourcetype=cisco_wsa_squid
| stats sum(bytessent) as Bytes by usage
| eval bandwidth = Bytes/(1024*1024)
```

usage	Bytes	bandwidth
Borderline	54011022	51.50892448425293
Business	66844576	63.747955322265625
Personal	299584913	285.7064371109009
Unknown	77187989	73.61220264434814
Violation	3203231	3.0548391342163086

eval Command – Round Values

- The results of Bandwidth are hard to read with so many decimal points
- round(*field/number*, *decimals*) function sets the value of a field to the number of decimals you specify
- In this example:
  - Divide the value of the Bytes field by (1024*1024)
  - Round the result to two decimal points
- If the number of decimals is unspecified, the result is a whole number

```
index=web sourcetype=access_combined
| stats sum(sc_bytes) as Bytes by usage
| eval bandwidth=round(Bytes/(1024*1024),2)
| sort -bandwidth
| rename bandwidth as "Bandwidth (MB)"
```

usage	Bytes	Bandwidth (MB)
Personal	299584913	285.71
Unknown	77187989	73.61
Business	66844576	63.75
Borderline	54011022	51.51
Violation	3203231	3.05

Removing Fields

- The "Bandwidth (MB)" field has the data in the desired format
- The Bytes field is no longer needed
  - The Bytes field can be removed

```
index=web sourcetype=access_combined
| stats sum(sc_bytes) as Bytes by usage
| eval bandwidth = round(Bytes/(1024*1024), 2)
| sort -bandwidth
| rename bandwidth as "Bandwidth (MB)"
| fields - Bytes
```

usage	Bandwidth (MB)
Personal	285.71
Unknown	73.61
Business	63.75
Borderline	51.51
Violation	3.05

eval Command – Calculating Values

You can perform mathematical functions against fields with numeric field values

In this example, stats calculates the total list price and total sale price by product_name

eval calculates the discount percentage and formats the

```
index=web sourcetype=access_combined
product_name=* action=purchase
| stats sum(price) as tp, sum(sale_price) as tsp by
product_name
| eval Discount = round(((tp - tsp)/ tp)*100)
| sort -Discount
| eval Discount = Discount."%"
| rename tp as "Total List Price", tsp as "Total Sale
  Price", product_name as Product
```

Product	Total List Price	Total Sale Price	Discount
Puppies vs. Zombies	578.84	230.84	60%
Fire Resistance Suit of Provolone	594.51	296.51	50%
Holy Blade of Gouda	742.76	370.76	50%
Dream Crusher	6438.39	4023.39	38%
Manganiello Bros.	5758.56	3598.56	38%

eval Command – tostring Function

- tostring converts a numeric field value to a string

  tostring(field,"*option*")

- Options:
  - "commas": applies commas
    - If the number includes decimals, it rounds to two decimal places
  - "duration": formats the number as "hh:mm:ss"
  - "hex": formats the number in hexadecimal

```
index=web sourcetype=access_combined
action=purchase status=503
| stats count(price) as NumberOfLostSales,
  avg(price) as AverageLostSales, sum(price) as
  TotalLostRevenue
| eval AverageLostSales
  = "$" + tostring(AverageLostSales, "commas"),
  TotalLostRevenue =$ +tostring(TotalLostRevenue,
                                "commas")
```

NumberOfLostSales	AverageLostSales	TotalLostRevenue
113	$21.58	$2,438.87

tostring Function – duration Option

This example shows "duration" option of tostring function

- stats calculates sessionTime for each session (JSESSIONID)
  - Use the range function to return the difference between the max and min values of _time
- sort 5 displays the top 5 most frequent values
- The duration option formats the time as "hh:mm:ss"

```
index=web sourcetype=access_combined
| stats range(_time) as sessionTime by JSESSIONID
| sort 5 -sessionTime
| eval duration =
tostring(sessionTime,"duration")
```

JSESSIONID	sessionTime	duration
SD6SL8FF3ADFF4951	175	00:02:55
SD6SL1FF10ADFF4966	158	00:02:38
SD6SL3FF1ADFF4959	141	00:02:21
SD0SL1FF2ADFF4963	135	00:02:15
SD4SL10FF2ADFF4961	132	00:02:12

eval Commands with Multiple Expressions

- Multiple expressions can be combined into one eval command
- Each subsequent expression references the results of previous expressions
- Expressions must be separated by commas

```
eval fieldname1 =
  expression1, fieldname2 =
  expression2,
      fieldname3 = expression3...
```

eval Command – if Function Syntax

if(*X*, *Y*, *Z*)

- The if function takes three arguments
- The first argument, *X*, is a Boolean expression
  - If it evaluates to TRUE, the result evaluates to the second argument, *Y*
  - If it evaluates to FALSE, the result evaluates to the third argument, *Z*
- Non-numeric values must be enclosed in "double quotes"
- Field values are treated in a case-sensitive manner

eval Command – if Function Example

- Create a new field, SalesTerritory
- Evaluate VendorID
  - If >= 7000 AND < 8000 is TRUE, set result to "Asia"
    - Remember, arguments must be enclosed in quotes
  - If it evaluates to FALSE, set result to "Rest of the World"

```
index=sales sourcetype=vendor_sales
| eval SalesTerritory
= if((VendorID >= 7000 AND VendorID < 8000), "Asia", "Rest of the
  World")
| stats sum(price) as TotalRevenue by SalesTerritory
| eval TotalRevenue = "$" + tostring(TotalRevenue, "commas")
```

SalesTerritory	TotalRevenue
Asia	$1,371.26
Rest of the World	$14,495.25

eval Command – case Function

case(X1,Y1,X2,Y2…)

Note

- The first argument, *X1*, is a Boolean expression
- If it evaluates to TRUE, the result evaluates to *Y1*
- If it evaluates to FALSE, the next Boolean expression, *X2*, is evaluated, etc.
- If you want an “otherwise” clause, just test for a condition you know is true at the end (e.g., 0=0)

```
index=web sourcetype=access_combined
| eval rating = case(productId LIKE "WC%", "Teen", productId LIKE "FS%", Mature",
0=0, "Unrated")
```

eval function

- To count the number of events that contain a specific field value, use the count and eval functions
  - Used within a transforming command, such as stats
  - Requires an as clause
  - Double quotes are required for character field values
  - Field values are case-sensitive

```
index=security sourcetype=linux_secure
vendor_action=*
| stats count(eval(vendor_action="Accepted")) as Accepted,
count(eval(vendor_action="Failed")) as Failed,
count(eval(vendor_action="session opened")) as SessionOpened
```

Accepted	Failed	SessionOpened
322	9676	407

Filtering Results – search and where

- The search and where commands both filter results
  - search
    - May be easier if you're familiar with basic search syntax
    - Treats field values in a case-insensitive manner
    - Allows searching on keyword
    - Can be used at any point in the search pipeline
  - where
    - Can compare values from two different fields
    - Functions are available, such as isnotnull()
    - Treats field values in a case-sensitive manner
    - Can't appear before first pipe in search pipeline

search Command

- To filter results, use search at any point in the search pipeline
- Behaves exactly like search strings before the first pipe
  - Uses the "*" wildcard
  - Treats field values in a case-insensitive manner

```
index=web sourcetype=access_combined
action=purchase status=200
| stats sum(price) as sales by product_name
| search sales>500
| sort -sales
| eval sales="$"+sales
| rename sales as "Popular Products",
  product_name as "Product Name"
```

Product Name	Popular Products
Manganiello Bros.	$839.79
Dream Crusher	$799.80
SIM Cubicle	$679.66

where Command

where eval-expression

- Uses same expression syntax as eval command
- Uses boolean expressions to filter search results and only keeps results that are True
- Double quoted strings are interpreted as field values
  - Treats field values in a case-sensitive manner
- Unquoted or single-quoted strings are treated as fields

`where` Command – Example

- Filters search results using eval expressions
- Used to compare two different fields

```
index=web sourcetype=access_combined
| timechart count(eval(action="changequantity"))
  as changes, count(eval(action="remove")) as
  removals
| where removals > changes
```

_time	changes	removals
2018-01-07	108	130
2018-01-09	126	135

where Command With like Operator

- Can do wildcard searches with where command
- Use (_) for one character and (%) for multiple characters
- Must use the like operator with wildcards

```
index=security sourcetype=linux_secure
| stats count by src_ip
| where src_ip like "10_.%"
```

src_ip	count
102.2.83.137	102
107.3.146.207	55
108.65.113.83	34
109.169.32.135	72

fillnull Command

- Use fillnull to replace null values in fields
- Use *value=string* to specify a string you want displayed instead

  Example: fillnull value=NULL
- If no value= clause, default replacement value is 0
- Optionally, restrict which fields fillnull apply to by listing them at end of command

  Example: fillnull VALUE="N/A" discount refund

fillnull Command – Examples

```
index=sales sourcetype=vendor_sales
| chart sum(price) over product_name by
VendorCountry
| fillnull
```

product_name	Morocco	United States
Dream Crusher	39.99	39.99
Puppies vs. Zombies	0	4.99

```
index=sales sourcetype=vendor_sales
| chart sum(price) over product_name by
VendorCountry
| fillnull value="No Value"
```

product_name	Morocco	United States
Dream Crusher	39.99	39.99
Puppies vs. Zombies	No Value	4.99

Correlating Events

What is a Transaction?

- A transaction is any group of related events that span time
- Events can come from multiple applications or hosts
  - Events related to a single purchase from an online store can span across an application server, database, and e-commerce engine
  - One email message can create multiple events as it travels through various queues
  - Each event in the network traffic logs represents a single user generating a single http request
  - Visiting a single website normally generates multiple http requests
    - HTML, JavaScript, CSS files
    - Flash, images, etc.

transaction Command

- transaction *field-list*
  - *field-list* can be one field name or a list of field names
  - Events are grouped into transactions based on the values of these fields
  - If multiple fields are specified and a relationship exists between those fields, events with related field values are grouped into a single transaction
- Common constraints:

  maxspan maxpause startswith endswith

Events That Have the Same JSESSIONID

- The log shows a number of events that share the same JSESSIONID value (SD0SL10FF3ADFF4950)
- However, it is difficult to:
  - View the events as a group
  - Gain insight as to what is happening with these events
  - Know if there are other events scattered in the results set

transaction Command – Example 1

- The transaction command creates a single event from a group of events
  - The events must share the same value in a specified field
- Transactions can cross multiple tiers such as web servers or application servers
- For example, you can easily view the events for JSESSIONID SD0SL10FF3ADFF4950

```
index=web sourcetype=access_combined
| transaction JSESSIONID
```

> 1/17/18 4:00:53.000 PM

175.44.1.122 - - [18/Jan/2018:00:00:53] "POST /category.screen?categoryId=SIMULATION&JSESSIONID=SD0SL10FF3ADFF4950 HTTP 1.1" 200 3526 "http://www.buttercupgames.com" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2; .NET CLR 1.1.4322; InfoPath.1; MS-RTC LM 8)" 222
175.44.1.122 - - [18/Jan/2018:00:01:05] "GET /category.screen?categoryId=STRATEGY&JSESSIONID=SD0SL10FF3ADFF4950 HTTP 1.1" 200 2453 "http://www.buttercupgames.com/oldlink?itemId=EST-18" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2; .NET CLR 1.1.4322; InfoPath.1; MS-RTC LM 8)" 989
175.44.1.122 - - [18/Jan/2018:00:01:08] "GET /oldlink?itemId=EST-6&JSESSIONID=SD0SL10FF3ADFF4950 HTTP 1.1" 200 3516 "http://www.buttercupgames.com/oldlink?itemId=EST-6" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2; .NET CLR 1.1.4322; InfoPath.1; MS-RTC LM 8)" 614
175.44.1.122 - - [18/Jan/2018:00:01:17] "GET /category.screen?categoryId=ACCESSORIES&JSESSIONID=SD0SL10FF3ADFF4950 HTTP 1.1" 200 652 "http://www.buttercupgames.com/oldlink?itemId=EST-16" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2; .NET CLR 1.1.4322; InfoPath.1; MS-RTC LM 8)" 118

transaction Command – Specific Fields

The transaction command produces additional fields, such as:

- duration – the difference between the timestamps for the first and last event in the transaction
- eventcount – the number of events in the transaction

transaction Command – maxspan/maxpause

- You can also define a max overall time span and max gap between events
  - maxspan=10m
    - Maximum total time between the *earliest* and *latest* events
    - If not specified, default is -1 (or no limit)
  - maxpause=1m
    - Maximum total time *between* events
    - If not specified, default is -1 (or no limit)

```
index=web sourcetype=access_combined
| transaction clientip maxspan=10m maxpause=1m
| eval duration = tostring(duration,"duration")
| sort -duration
| table clientip duration action
| rename clientip as "Client IP", action as
  "Client Actions"
```

Client IP	duration	Client Actions
91.199.80.24	00:01:27	addtocart changequantity purchase view
12.130.60.5	00:01:27	addtocart purchase view
60.220.218.88	00:01:25	view

transaction Command – startswith/endswith

- To form transactions based on terms, field values, or evaluations, use startswith and endswith options
- In this example:
  - The first event in the transaction includes addtocart
  - The last event includes purchase

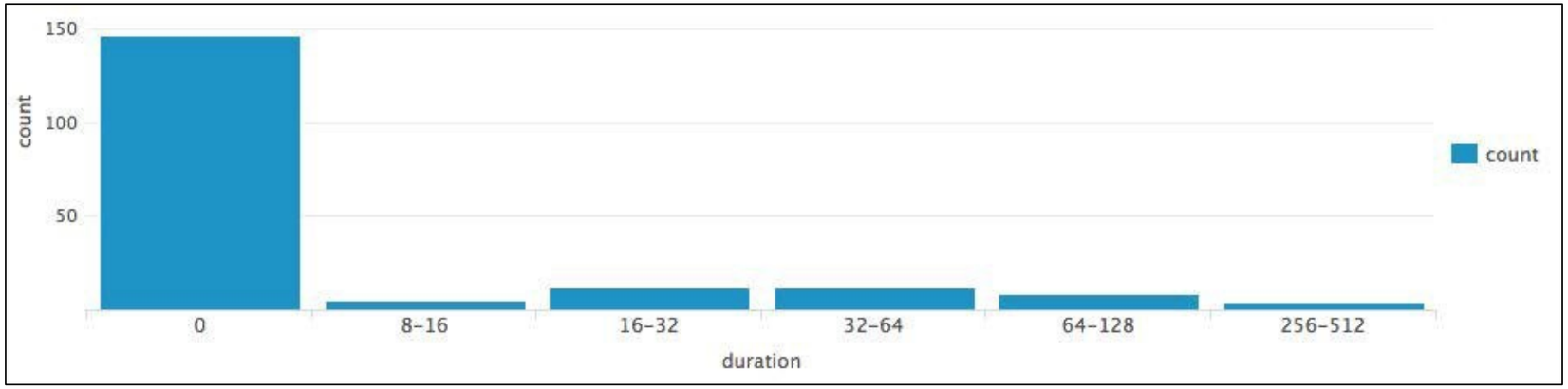
```
index=web sourcetype=access_combined
| transaction clientip JSESSIONID
  startswith=eval(action="addtocart")
  endswith=eval(action="purchase")
| table clientip, JSESSIONID, duration, eventcount
```

clientip	JSESSIONID	duration	eventcount
223.205.219.198	SD1SL9FF5ADFF4953	3	2
201.3.120.132	SD10SL5FF2ADFF4956	4	2
201.3.120.132	SD10SL5FF2ADFF4956	1	2
203.45.206.135	SD10SL7FF4ADFF4964	2	2

Reporting on Transactions

- You can use statistics and reporting commands with transactions
- `count()` function counts number of transactions and separates the count by the duration of each

```
index=web sourcetype=access_combined
status=200 action=purchase
| transaction clientip maxspan=10m
| chart count BY duration span=log2
```



transaction vs. stats

- When you have a choice, use `stats` — it's faster and more efficient, especially in large Splunk environments
- Only use `transaction` when you:
  - Need to see events correlated together
  - Must define event grouping based on `start/end` values or segment on time
- Use `stats` when you:
  - Want to see the results of a calculation
  - Can group events based on a field value (e.g., `by src_ip`)
- By default, there's a limit of 1,000 events per `transaction`
  - No such limit applies to `stats`
  - Admins can change limit by configuring `max_events_per_bucket` in `limits.conf`

transaction vs. stats: Example 2

```
index=security
sourcetype=linux_secure failed
| transaction src_ip
| table src_ip, eventcount
| sort - eventcount
```

src_ip	eventcount
87.194.216.51	1000
211.166.11.101	840
128.241.220.82	662
194.215.205.19	617
87.194.216.51	134
236.45.28.248	133
229.156.63.239	133

- Transaction took 6.163 seconds
- Stats took 4.643 seconds
- transaction has a limit of 1,000
- Count of transactions vs. count of IPs

```
index=security
sourcetype=linux_secure failed
| stats count as
  eventcount by src_ip
| sort - eventcount
```

src_ip	eventcount
87.194.216.51	1134
211.166.11.101	840
128.241.220.82	662
194.215.205.19	617
109.169.32.135	443
216.221.226.11	432
188.138.40.166	429

Knowledge Objects

Module Objectives

- Identify the categories of knowledge objects
- Define the role of a knowledge manager
- Identify naming conventions
- Review permissions
- Manage knowledge objects
- Describe the Splunk Common Information Model (CIM)

What are Knowledge Objects?

- Knowledge objects are tools you use to discover and analyze various aspects of your data
  - Data interpretation – Fields and field extractions
  - Data classification – Event types
  - Data enrichment – Lookups and workflow actions
  - Normalization – Tags and field aliases
  - Datasets – Data models

What are Knowledge Objects? (cont.)

- Shareable
  - Can be shared between users
- Reusable
  - Persistent objects that can be used by multiple people or apps, such as macros and reports
- Searchable
  - Since the objects are persistent, they can be used in a search

Defining Naming Conventions

- This course uses simple names for lab exercises, but using a naming convention in your production environment is recommended. For example:
  - **Group:** Corresponds to the working group(s) of the user saving the object (examples: SEG. NEG. OPS. NOC)
  - **Object Type:** Indicates the type of object (alert, report, summary-index-populating) (examples: Alert, Report, Summary)
  - **Description:** A meaningful description of the context and intent of the search, limited to one or two words if possible; ensures the search name is unique
- So, for example:
  SEG_Alert_WinEventlogFailures

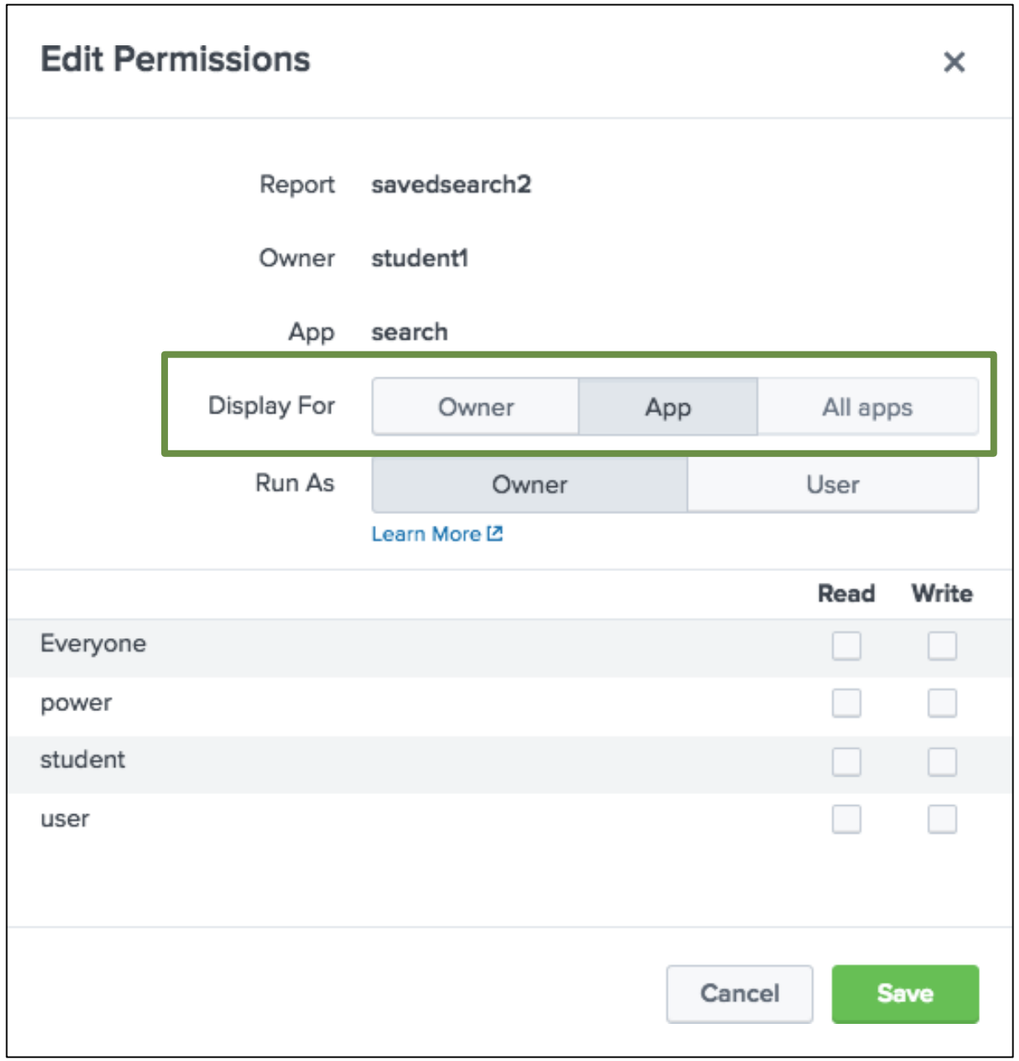
Reviewing Permissions

Description		Create	Read	Edit (write)
Private	Only the person who created the object can use it and edit it	User Power Admin		Person who created it Admin
This app only	Object persists in the context of a specific app	Power Admin	User* Power* Admin	User* Powe* Admin
All apps	Object persists globally across all apps	Admin	User* Power* Admin	User* Power* Admin

* Permission to read and/or write if creator gives permission to that role

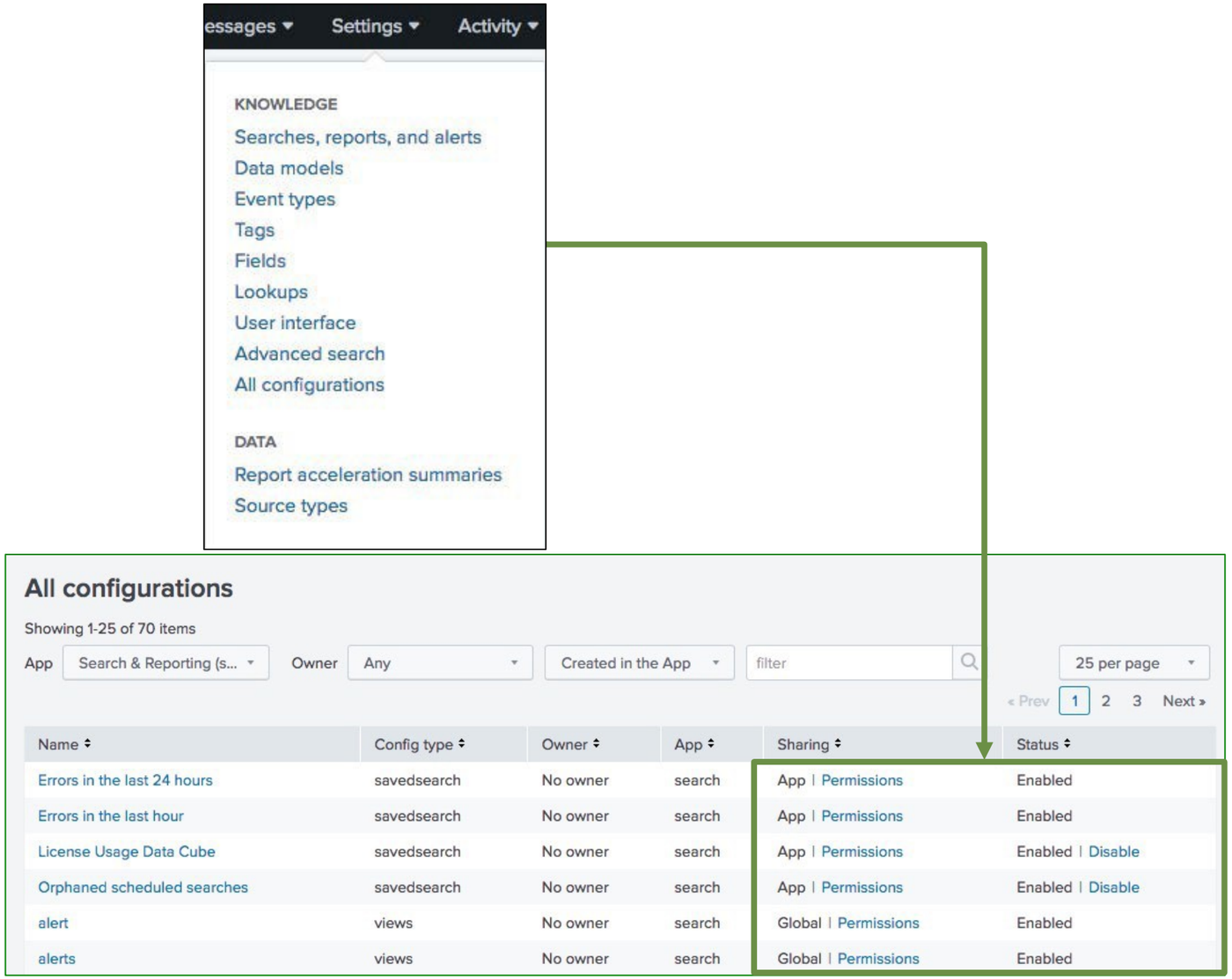
Reviewing Permissions (cont.)

- When an object is created, Display For is set to **Owner** by default
- When object's permissions are set to **App** or **All apps**, all roles are given read permission
  - Write permission is reserved for admin and the object creator unless the creator edits permissions
- Only the admin role can promote an object to **All apps**
  - Other roles have **All Apps** button grayed out



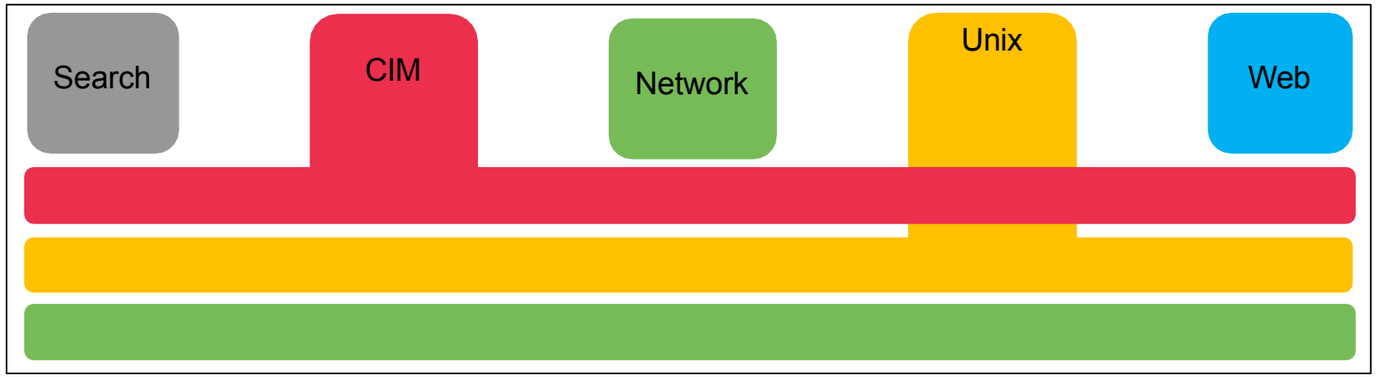
Managing Knowledge Objects

- Knowledge objects are centrally managed from **Settings > Knowledge**
- Your role and permissions determine your ability to modify an object's settings



Using the Splunk Common Information Model (CIM)

- Methodology for normalizing data
- Easily correlate data from different sources and source types
- Leverage to create various objects discussed in this course—field extractions, field aliases, event types, tags
- More details discussed in Module 13



Creating and Managing Fields

Field Auto-Extraction

- Splunk automatically discovers many fields based on source type and key/value pairs found in the data
- Prior to search time, some fields are already stored with the event in the index
  - Meta fields, such as `host`, `source`, and `sourcetype`
  - Internal fields such as `_time` and `_raw`
- At search time, *field discovery* discovers fields directly related to the search's results
- Splunk may also extract other fields from raw event data that aren't directly related to the search

Performing Field Extractions

- In addition to the many fields Splunk auto-extracts, you can also extract your own fields with the Field Extractor (FX)
- Use FX to extract fields that are static and that you use often in searches
  - Graphical UI
  - Extract fields from events using regex or delimiter
  - Extracted fields persist as knowledge objects
  - Can be shared and re-used in multiple searches
- Access FX via Settings, Fields Sidebar, or Event Actions menu

Field Extraction Methods

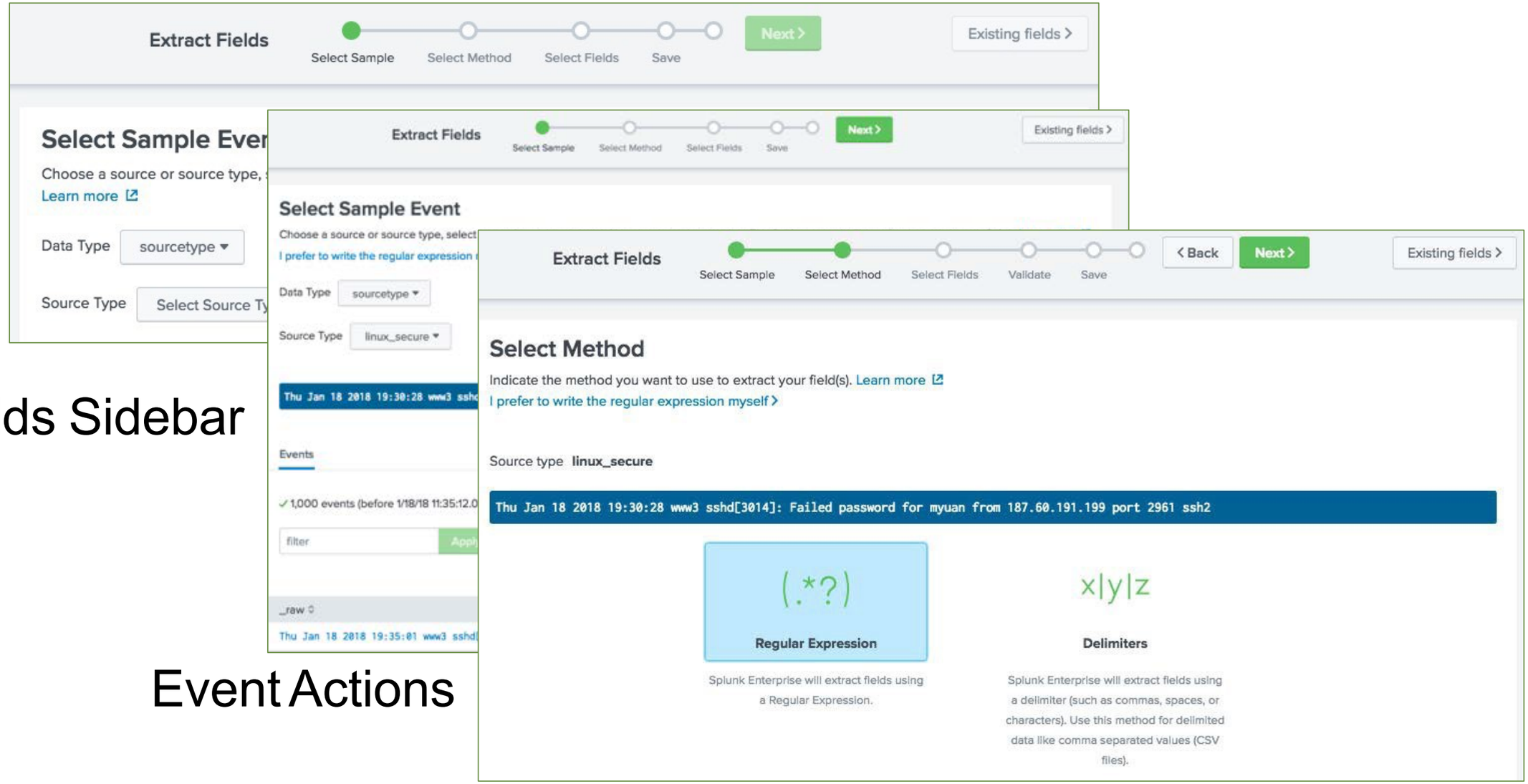
- Regex
  - Use this option when your event contains unstructured data like a system log file
  - FX attempts to extract fields using a Regular Expression that matches similar events
- Delimiter
  - Use this option when your event contains structured data like a .csv file
  - The data doesn’t have headers and the fields must be separated by delimiters (spaces, commas, pipes, tabs, or other characters)

Field Extraction Workflows – RegEx

Settings

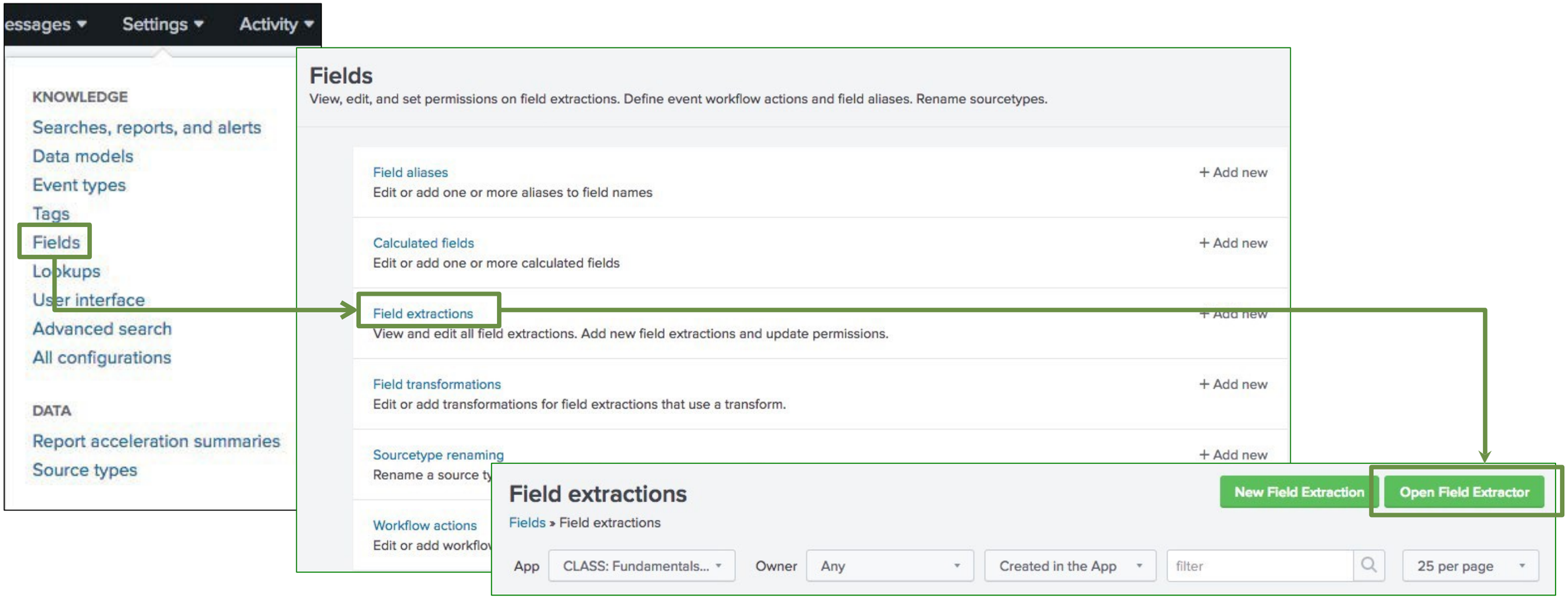
Fields Sidebar

Event Actions



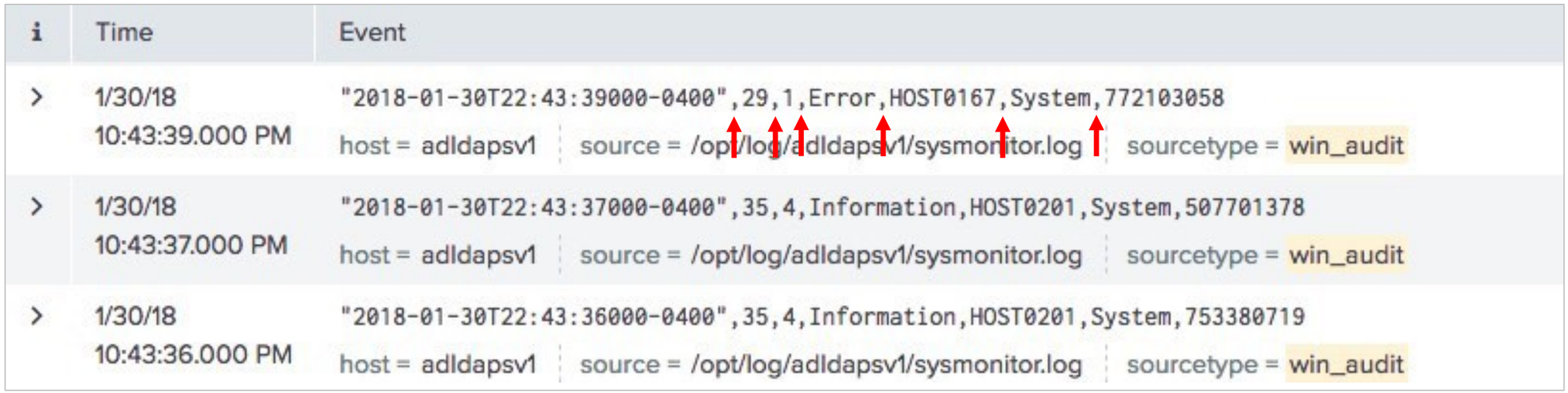
Regex Field Extractions from Settings

Settings > Fields > Field extractions > Open Field Extractor



Delimited Field Extractions

- Use delimited field extractions when the event log does not have a header and fields are separated by spaces, commas, or characters
- In this example, the fields are separated by commas

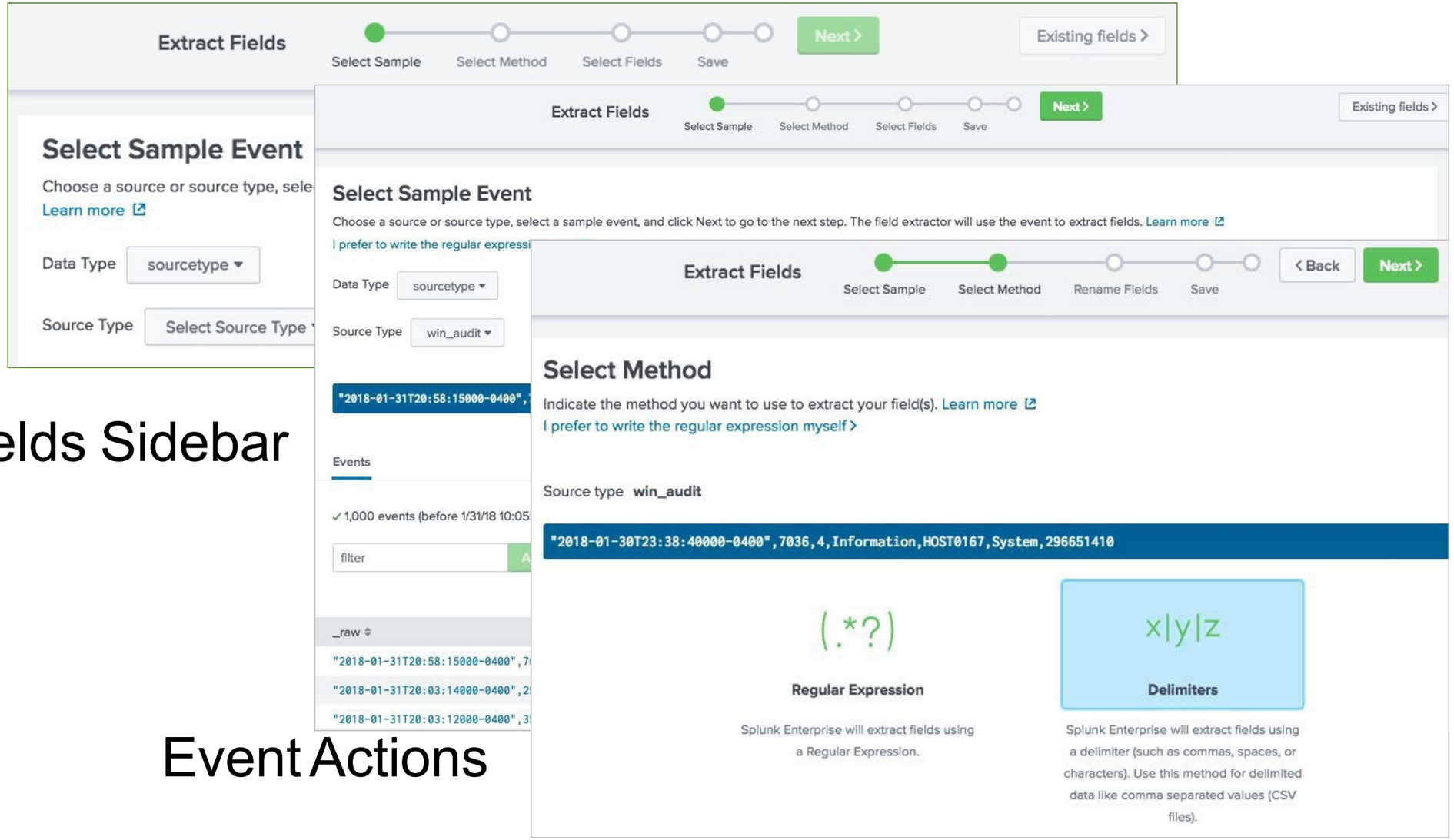


Field Extraction Workflows – Delimiters

Settings

Fields Sidebar

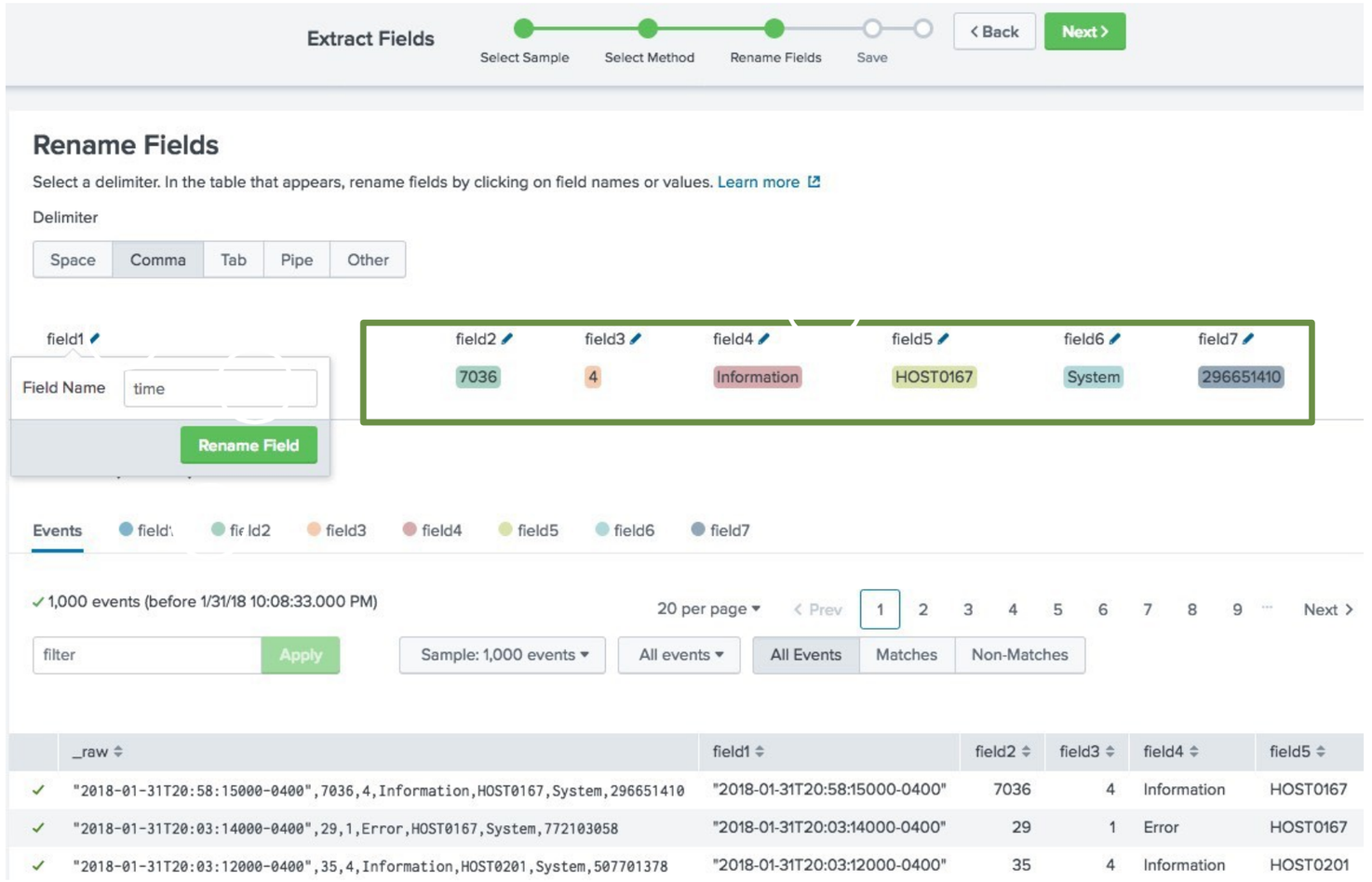
Event Actions



Delimited Field Extractions (Settings) – Rename Field

Click the icon next to the default field name

Enter a new field name



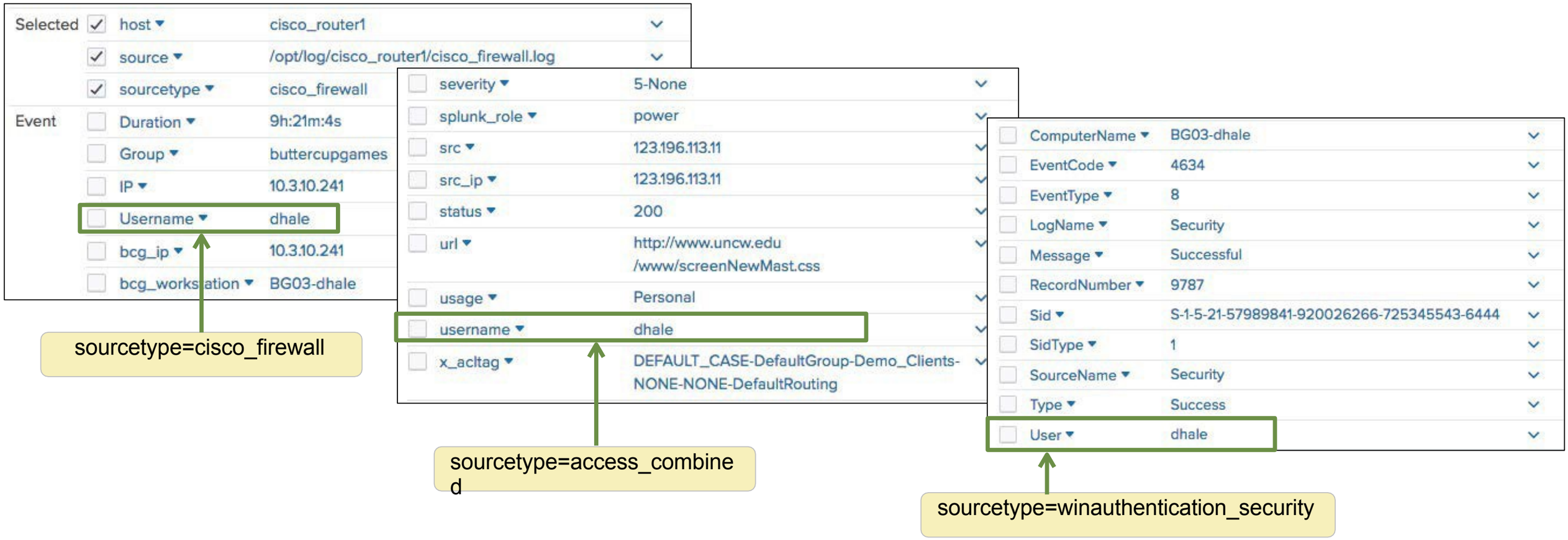
Creating Field Aliases and Calculated Fields

Field Aliases

- A way to normalize data over any default field (host, source or sourcetype)
- Multiple aliases can be applied to one field
- Applied after field extractions, before lookups
- Can apply field aliases to lookups

Field Alias Example

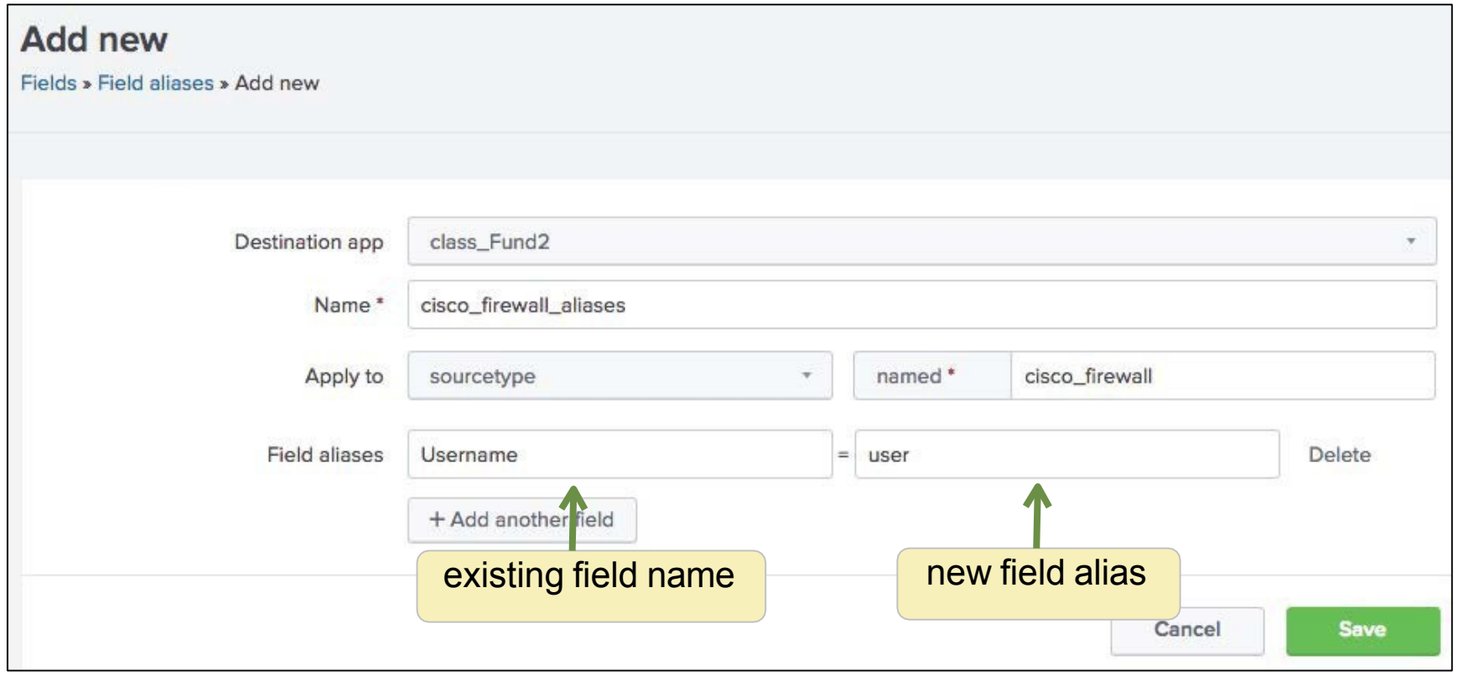
- Several source types contain some type of a username field
- To make data correlation and searching easier, normalize the username field



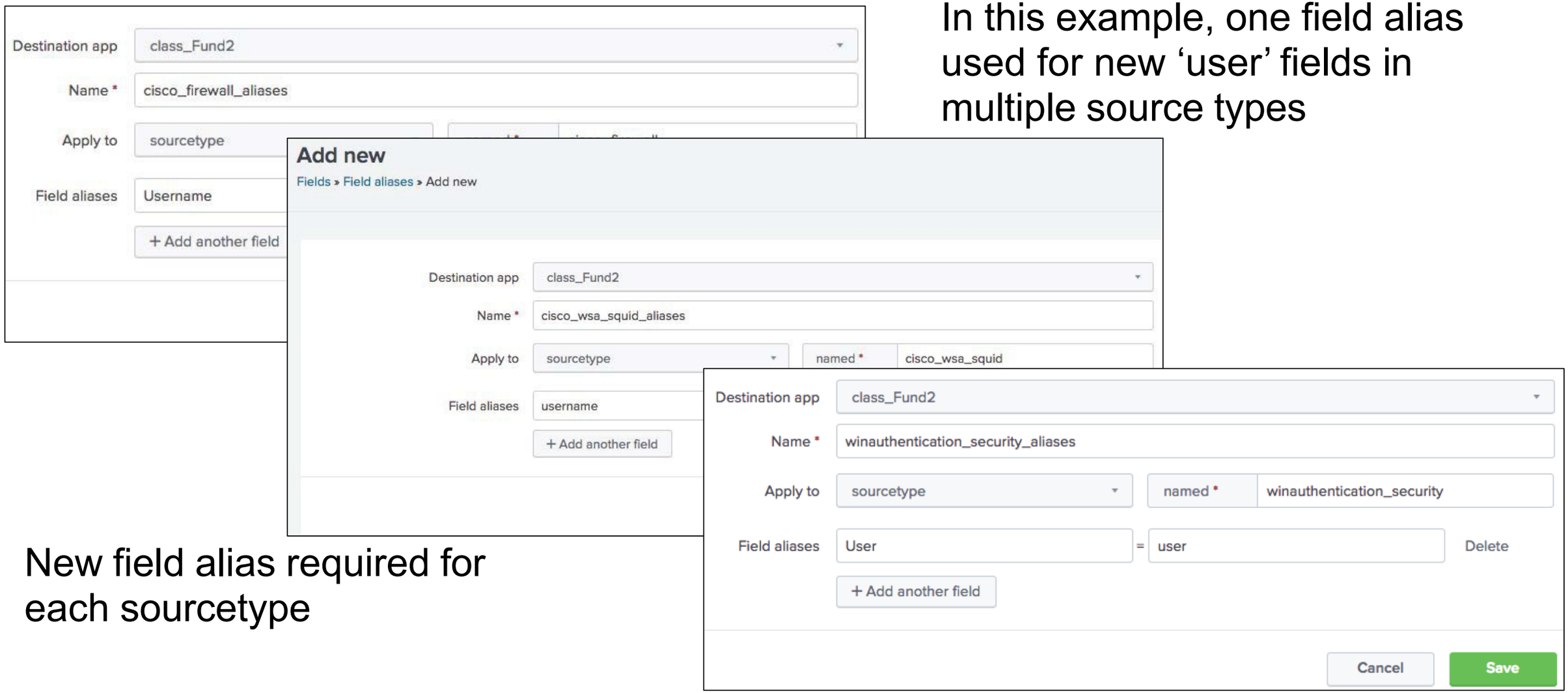
Creating a Field Alias

Settings > Fields > Field Aliases > New Field Alias

1. Select the app associated with the field alias
2. Enter a Name for the field alias
3. Apply the field alias to a default field:
   - Host
   - Source
   - Sourcetype
4. Enter the name for the existing field and the new alias



Creating a Field Alias (cont.)

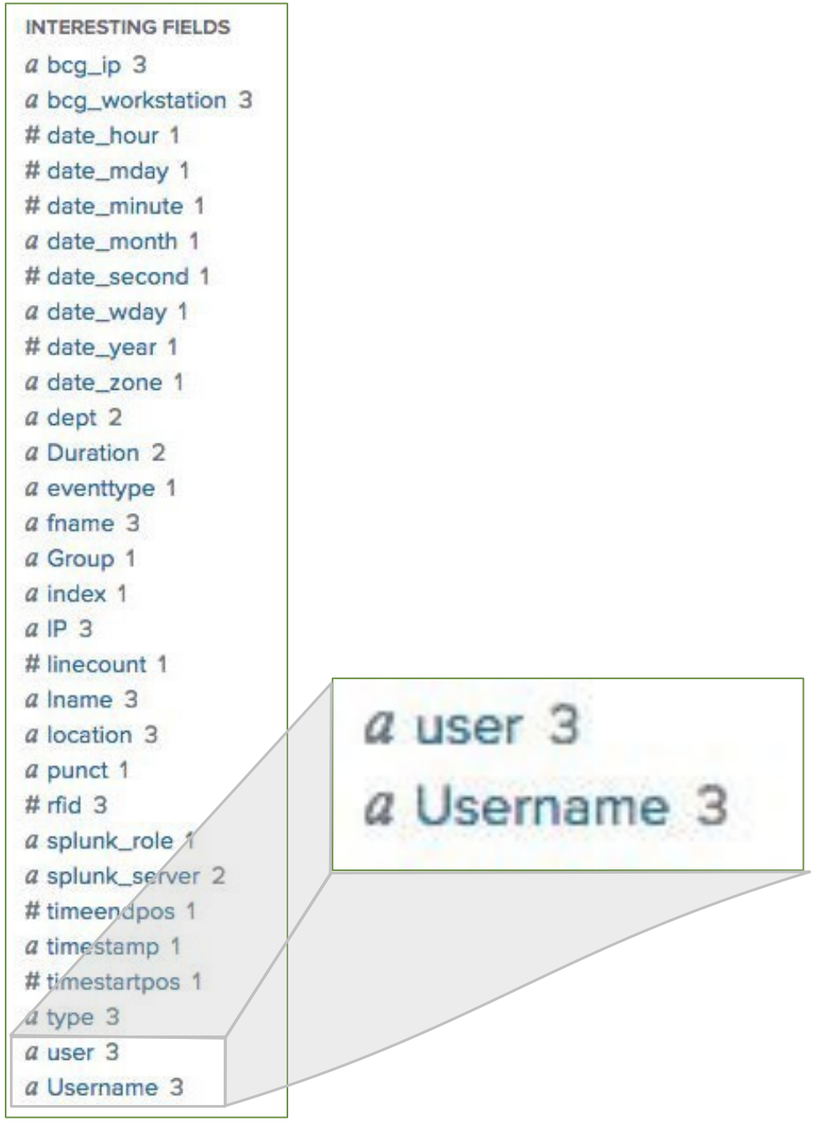


In this example, one field alias used for new ‘user’ fields in multiple source types

New field alias required for each sourcetype

Field Alias and Original Fields

- When you create a field alias, the original field is not affected
- Both fields appear in the All Fields and Interesting Fields lists, if they appear in at least 20% of events



Field Aliases and Lookups

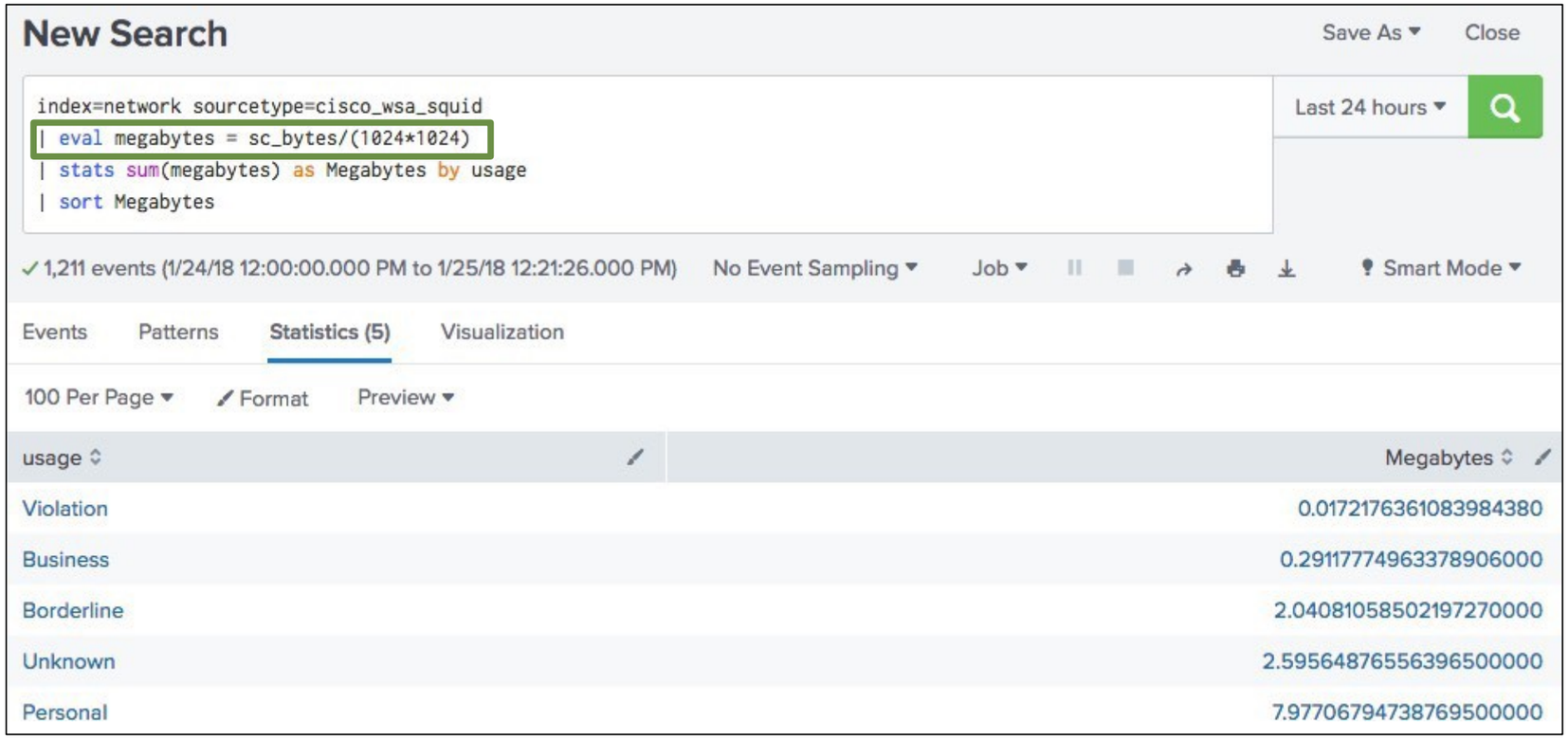
After you have defined your field aliases, you can reference them in a lookup table

employees.csv

```
rfid,fname,lname,user,email,dept,location,ip
108423575302,Alle,Pucci,apucci,apucci@buttercupgames.com,Sales,Boston,10.3.10.53
672903009231,Dwight,Hale,dhale,dhale@buttercupgames.com,Sales,Boston,10.3.10.241
398009643042,Phyllis,Bunch,pbunch,pbunch@buttercupgames.com,ITOps,Boston,10.3.10.227
374765319282,Enrique,Maxwell,emaxwell,emaxwell@buttercupgames.com,ITOps,Boston,10.3.10.46
227128834140,David,Johnson,djohnson,djohnson@buttercupgames.com,Engineering,Boston,10.3.10.180
371211812887,Galina,Zuyeva,gzuyeva,gzuyeva@buttercupgames.com,Engineering,Boston ,10.3.10.67
249772079712,Louis,Sagers,lsagers,lsagers@buttercupgames.com,SecOps,Boston,10.3.10.21
,,,,,,,
417852300683,Amanda,Curry,acurry,acurry@buttercupgames.com,SecOps,San Francisco,10.1.10.252
542830538161,Alan,Dombrowski,adombrowski,adombrowski@buttercupgames.com,SecOps,San Francisco,10.1.10.129
768166372290,Cerys,Farrell,cfarrell,cfarrell@buttercupgames.com,Sales,San Francisco,10.1.10.107
153218951159,Placido,Toscani,ptoscani,ptoscani@buttercupgames.com,Sales,San Francisco,10.1.10.38
994499284304,Ian,King,iking,iking@buttercupgames.com,Sales,San Francisco,10.1.10.201
531253083348,Gabriel,Voronoff,gvoronoff,gvoronoff@buttercupgames.com,Marketing,San Francisco,10.1.10.163
520156890727,Bao,Lu,blu,blu@buttercupgames.com,Marketing,San Francisco,10.1.10.100
727896988001,Lien,Teng,lteng,lteng@buttercupgames.com,ITOps,San Francisco,10.1.10.15
936901629743,Gabriel,Voronoff,gvoronoff,gvoronoff@buttercupgames.com,ITOps,San Francisco,10.1.10.163
230876363319,Meng,Yuan,myuan,myuan@buttercupgames.com,Engineering,San Francisco,10.1.10.172
271108583080,Patrick,Callahan,pcallahan,pcallahan@buttercupgames.com,Engineering,San Francisco,10.1.10.98
569361105570,Kathleen,Percy,kpercy,kpercy@buttercupgames.com,Compliance Officer,San Francisco,10.1.10.216
,,,,,,,
145297537706,Nigella,Pearce,npearce,npearce@buttercupgames.com,SecOps,London,10.2.10.70
632071692298,Yanto,Owen,yowen,yowen@buttercupgames.com,Sales,London,10.2.10.170
862417886973,Finlay,Bryan,fbryan,fbryan@buttercupgames.com,Sales,London,10.2.10.166
890313901800,Bradley,Hussain,bhussain,bhussain@buttercupgames.com,ITOps,London,10.2.10.22
425932411002,Naomi,Sharpe,nsharpe,nsharpe@buttercupgames.com,ITOps,London,10.2.10.163
```

What is a Calculated Field?

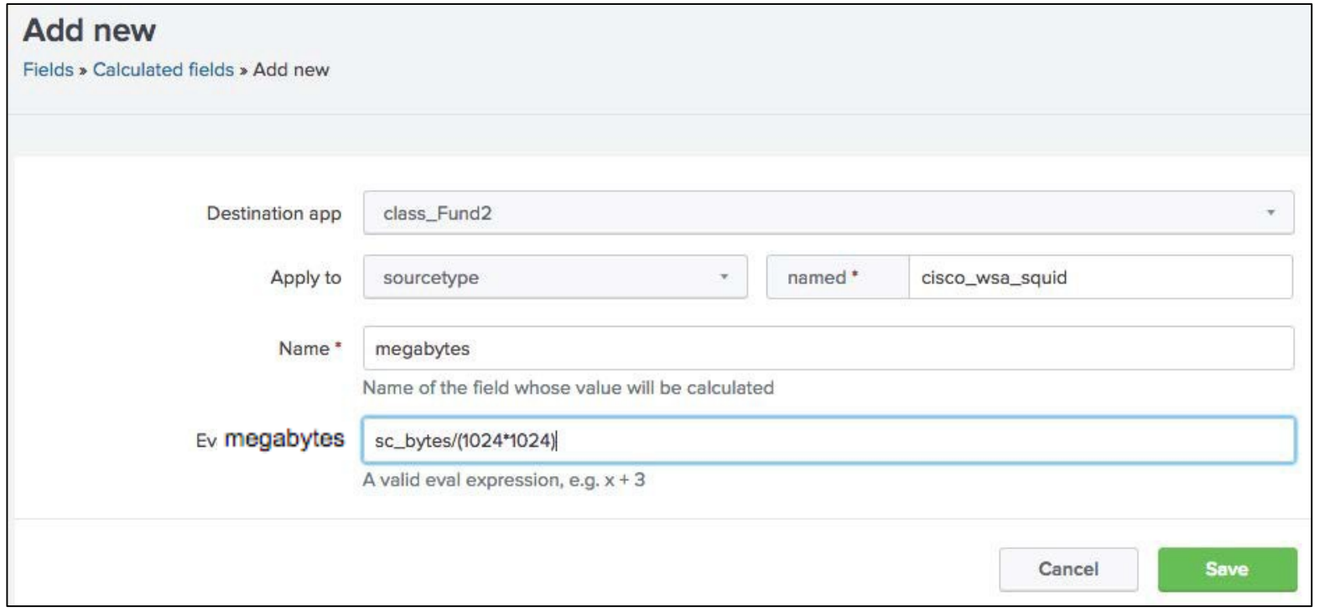
- Shortcut for performing repetitive, long, or complex transformations using the `eval` command
- Must be based on an extracted field



Creating a Calculated Field

Settings > Fields > Calculated Fields > New Calculated Field

1. Select the app that will use the calculated field
2. Select host, source, or sourcetype to apply to the calculated field and specify the related name
3. Name the calculated field
4. Define the eval expression



Working with Tags and Event Types

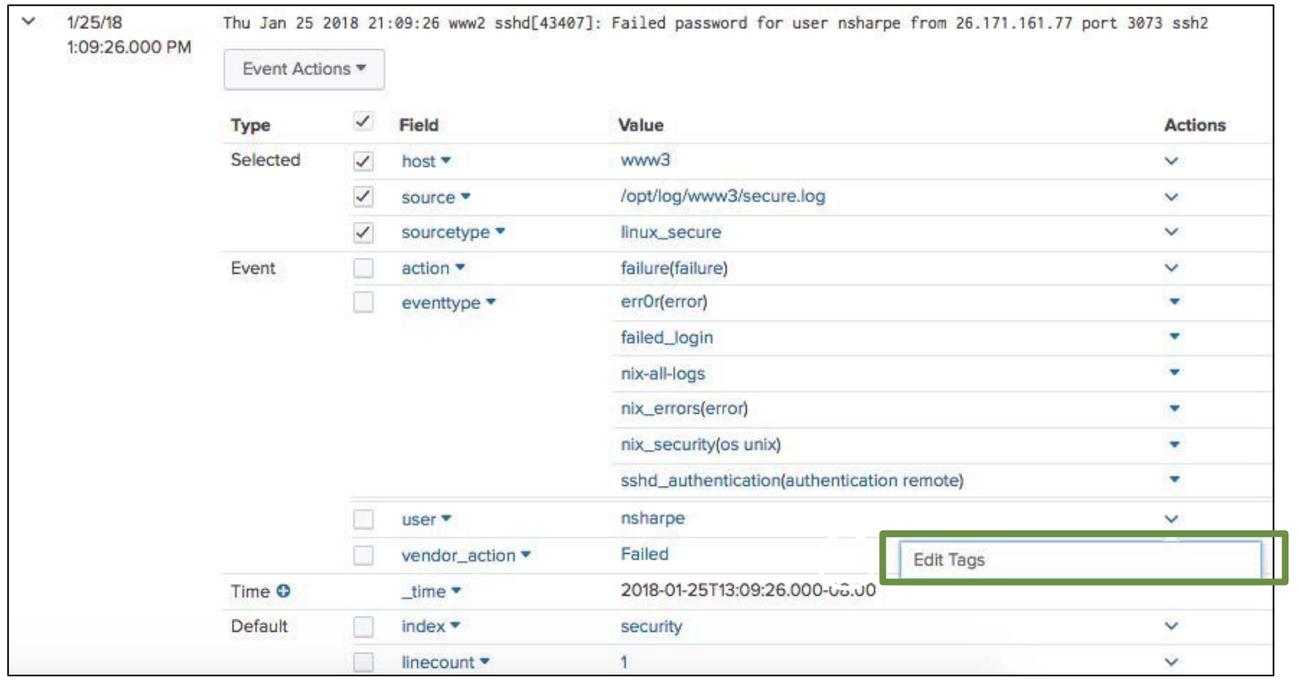
Describing Tags

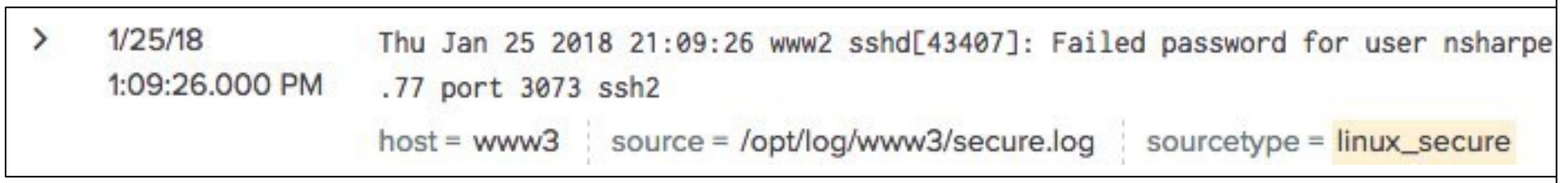
- Tags are like nicknames that you create for related field/value pairs
- Tags make your data more understandable and less ambiguous
- You can create one or more tags for any field/value combination
- Tags are case sensitive

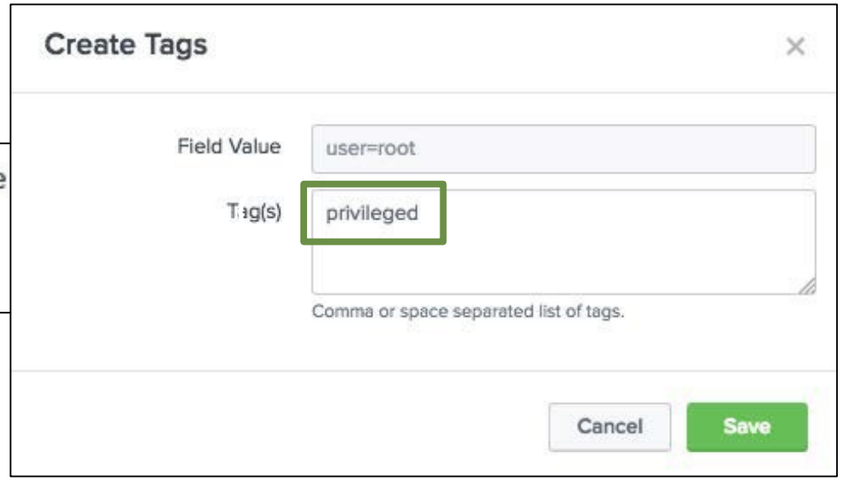
Creating Tags

To create a tag:

1. Click on the arrow for event details
2. Under Actions, click the down arrow
3. Select Edit Tags
4. Name the tags, separated by commas

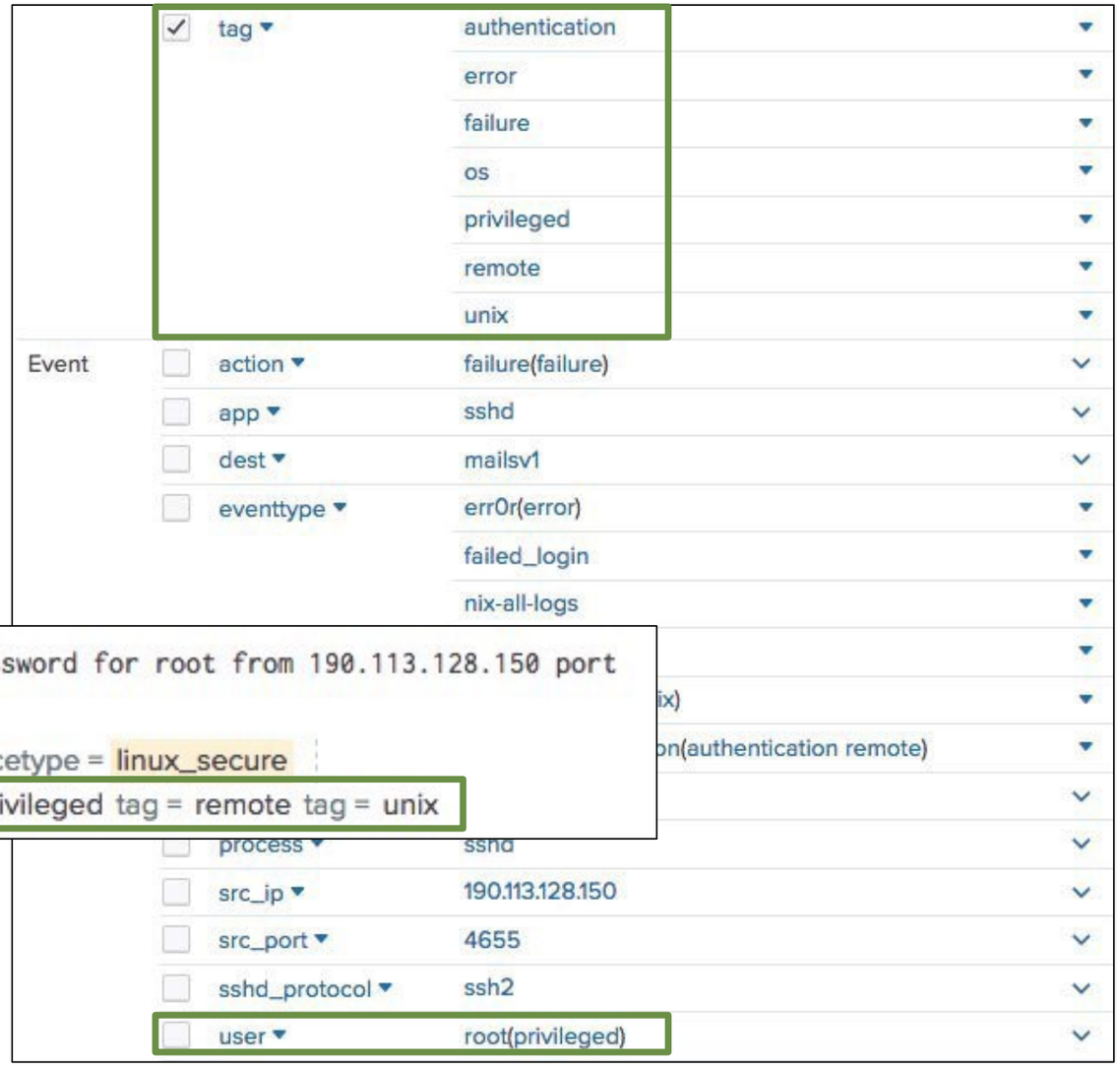






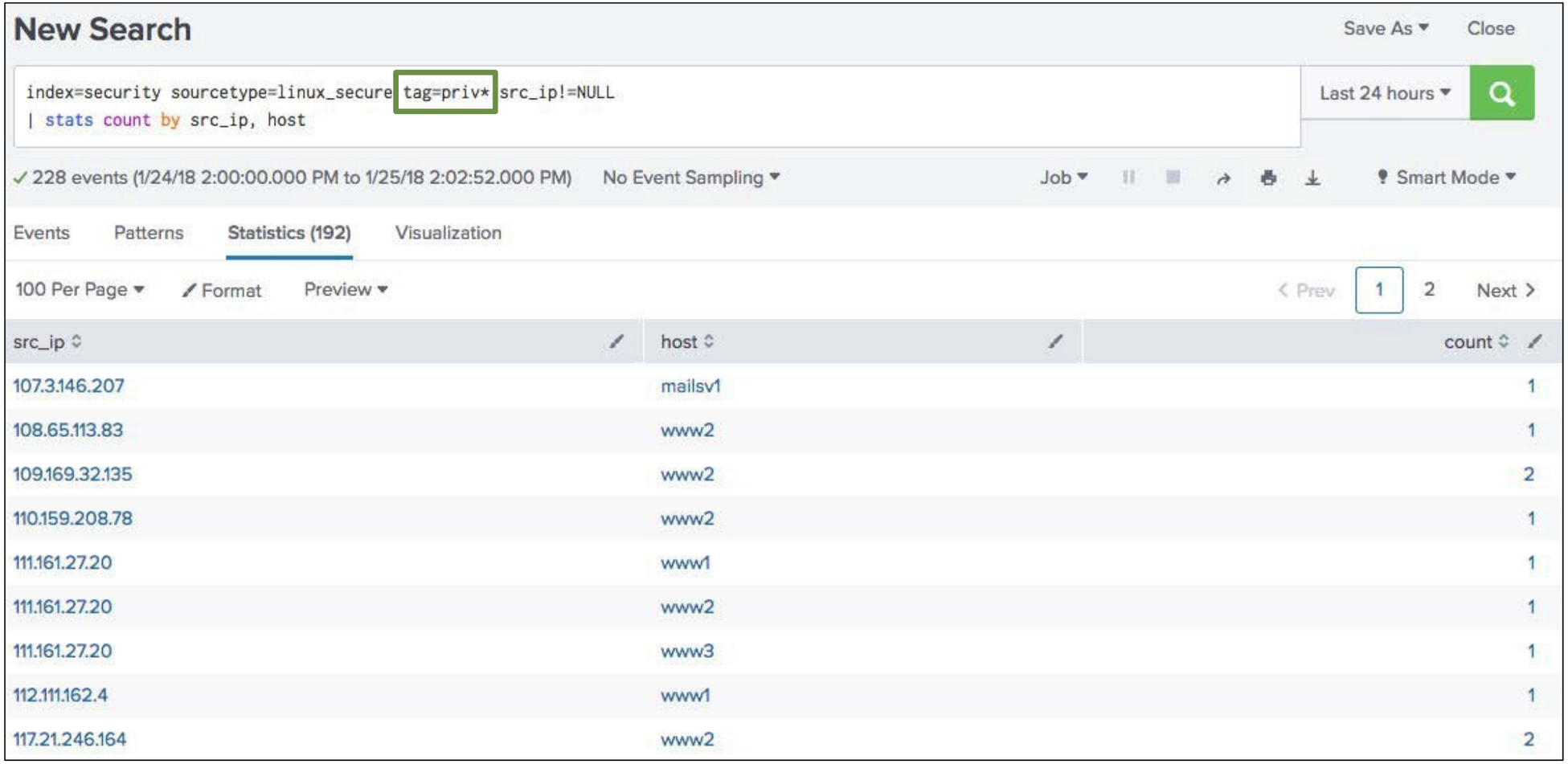
Viewing Tags

- When tagged field/value pairs are selected, the tags appear:
  - In the results as tags
  - In parentheses next to the associated field/value pairs



Using Tags

To use tags in a search, use the syntax: **tag=<*tag name*>**



Searching for Tags

- To search for a tag associated with a value:
  - tag=<tagname>

```
tag=privileged
```

- To search for a tag associated with a value on a specific field:
  - tag::<field>=<tagname>

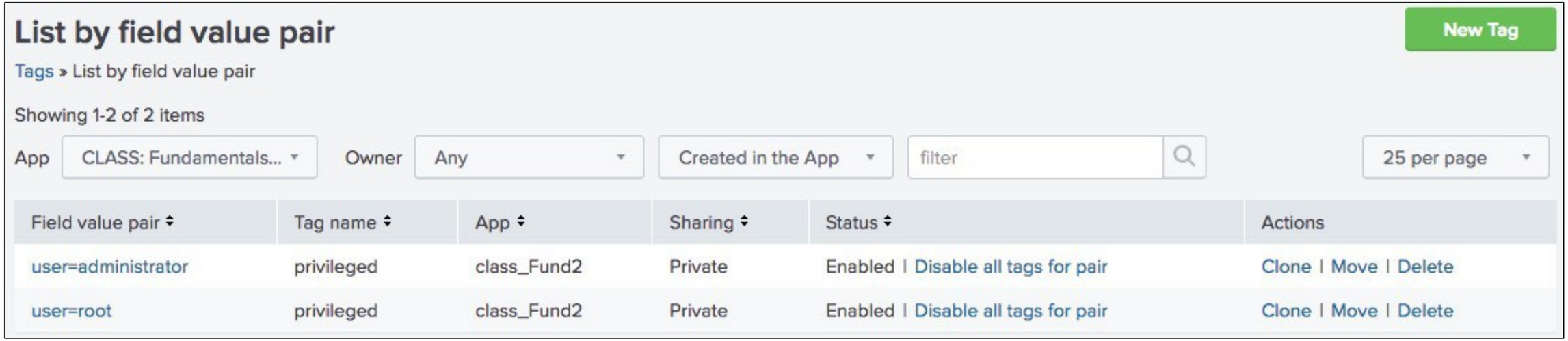
```
tag::user=privileged
```

- To search for a tag using a partial field value:
  - Use (*) wildcard

```
tag=p*
```

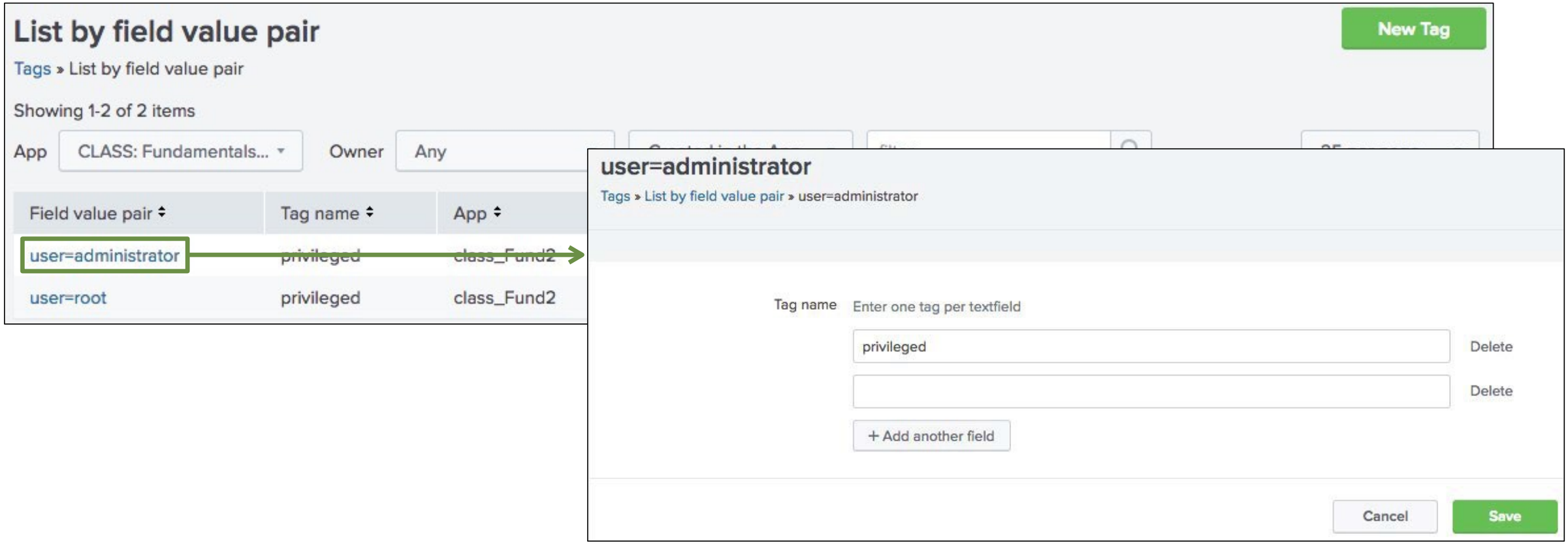
Managing Tags – List by Field Value Pair

- **Settings > Tags > List by field value pair**
  - Edit permissions
  - Disable all tags for pair – disables the tag in searches and prevents it from being listed under List by Tag Name and All unique tag objects



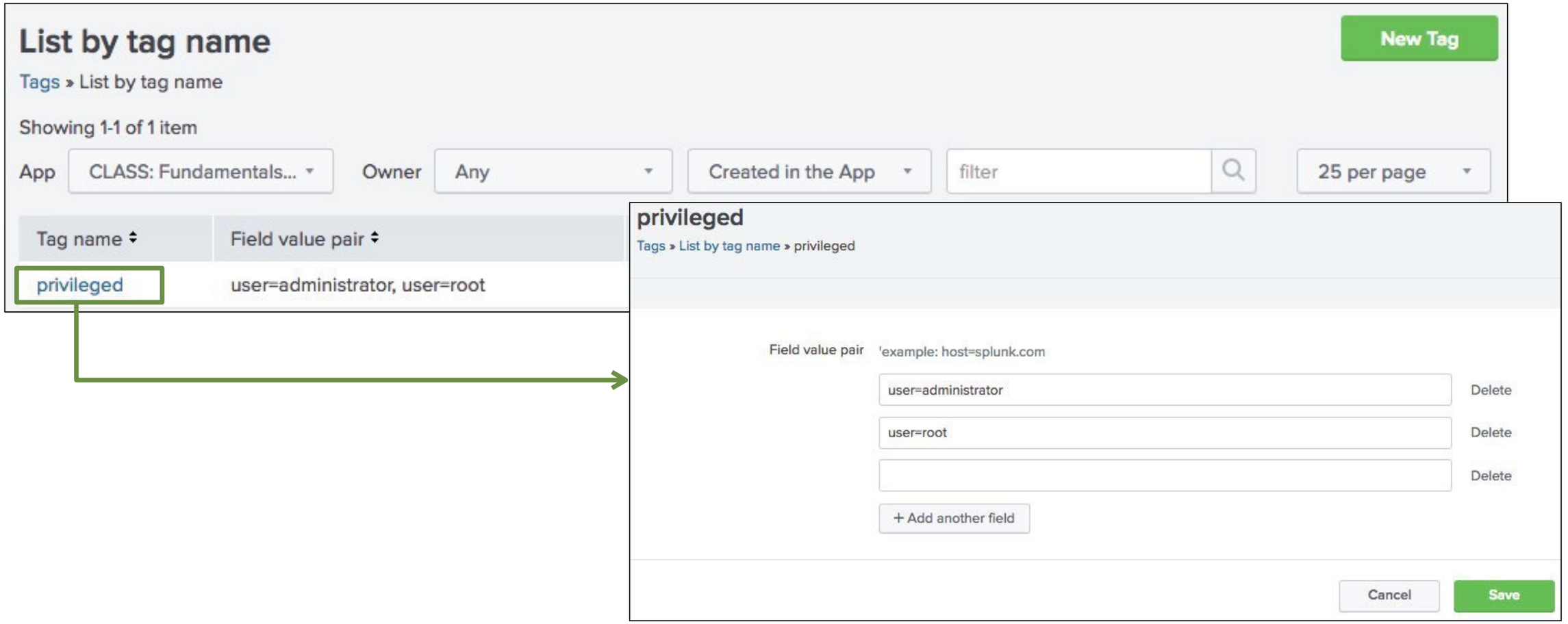
Adding/Changing the Tag Name

Click **List by field value pair** to add another tag or change the name of the tag



Adding/Changing the Field Value Pair

Click **List by tag name** to add or edit the field value pair for the tag



Describing Event Types

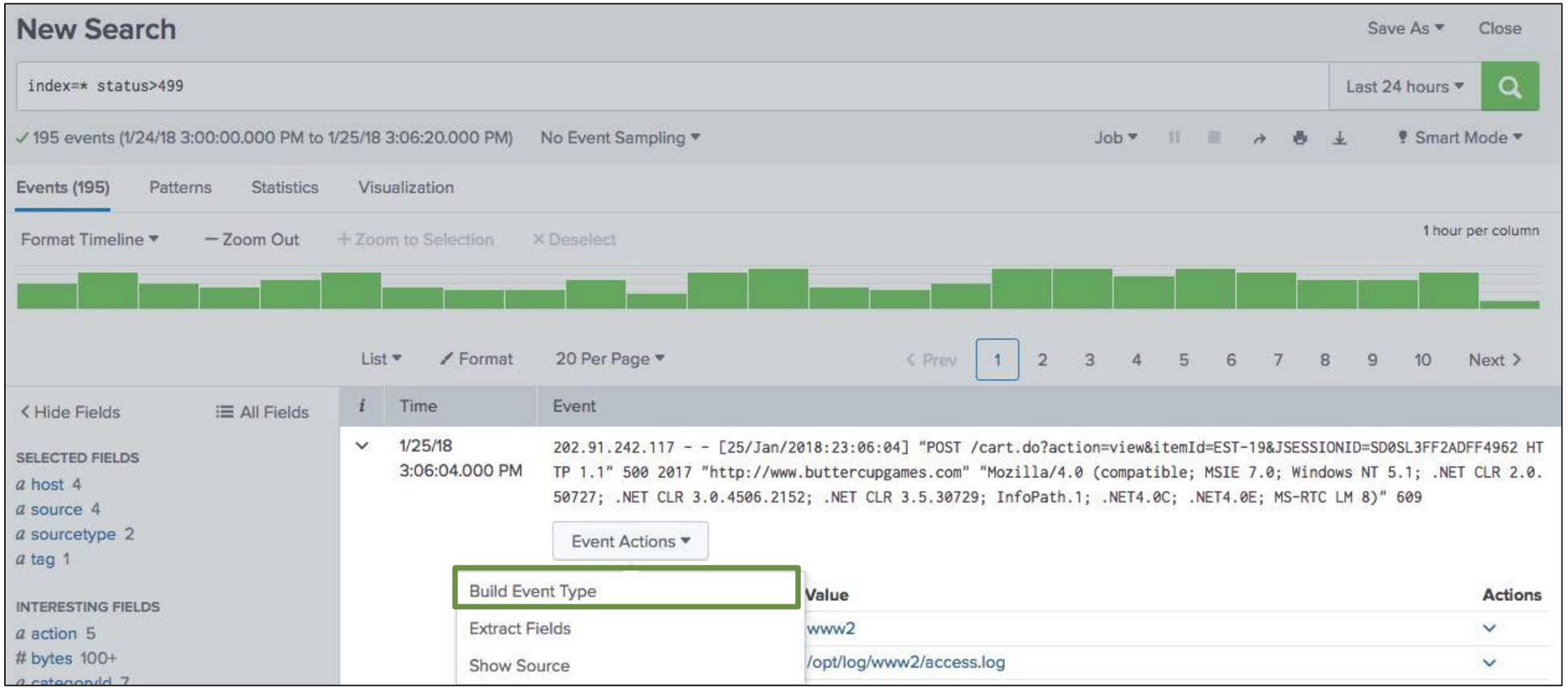
- A method of categorizing events based on a search
- A useful method for institutional knowledge capturing and sharing
- Can be tagged to group similar types of events

Creating an Event Type from the Search Page

1. Run a search and verify that all results meet your event type criteria
2. From the Save As menu, select **Event Type**
3. Provide a Name for your event type (name should not contain spaces)

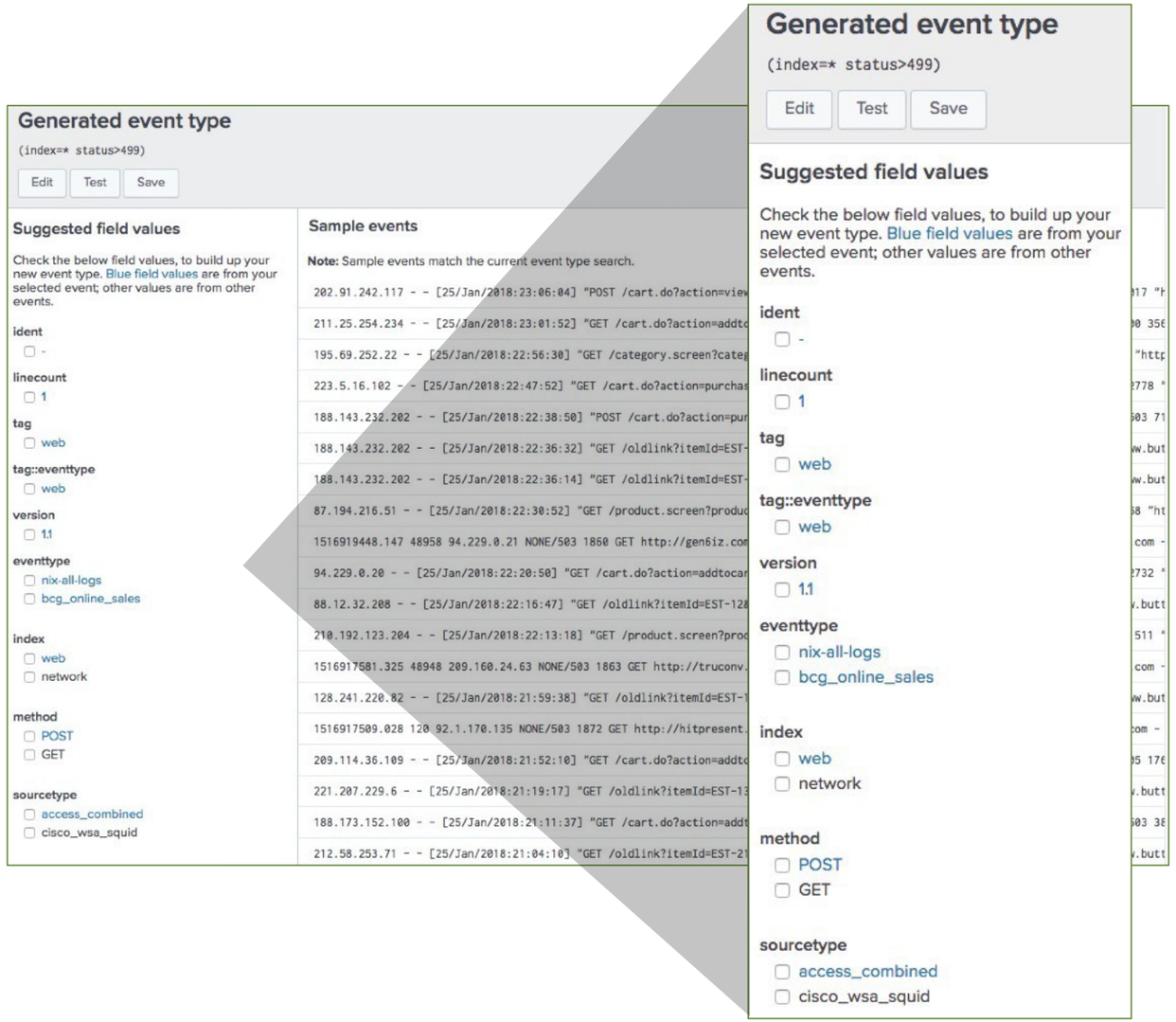
Using the Event Type Builder

1. From the event details, select **Event Actions > Build Event Type**



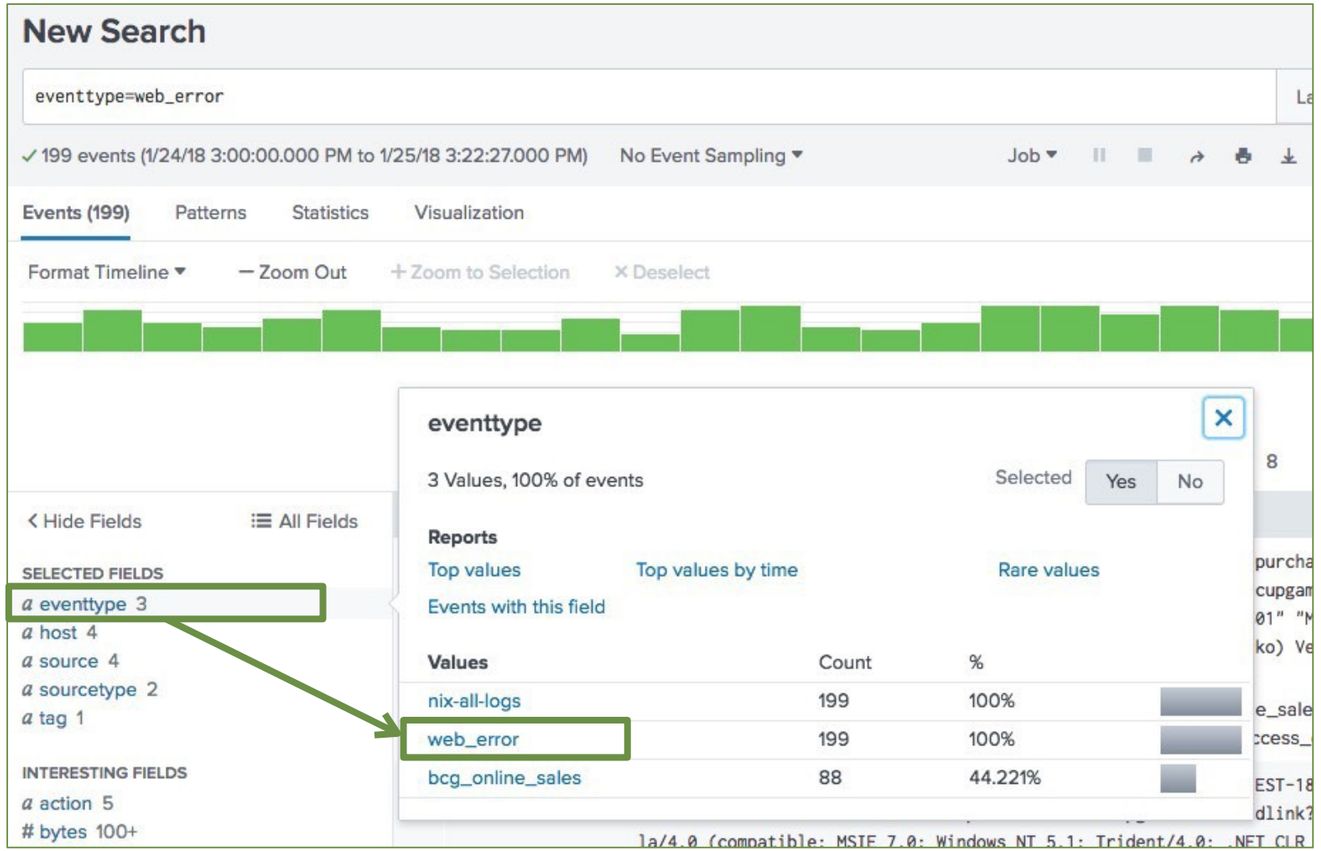
Using the Event Type Builder (cont.)

2. Refine the criteria for your event type such as:
   - Search string
   - Field values
   - Tags
3. Verify your selections and click Save



Using Event Types

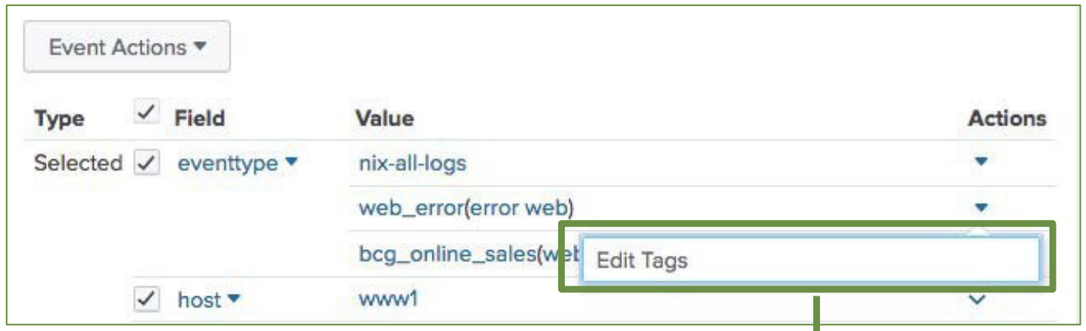
- To verify the event type, search for `eventtype=web_error`
- 'eventtype' displays in the Fields sidebar and can be added as a selected field
- Splunk evaluates the events and applies the appropriate event types at search time
- Using the Fields sidebar, you can easily view the individual event types, the number of events, and percentage

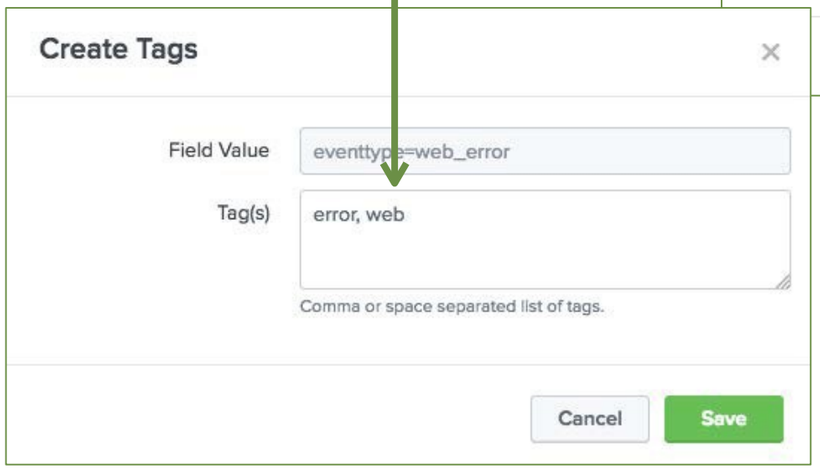


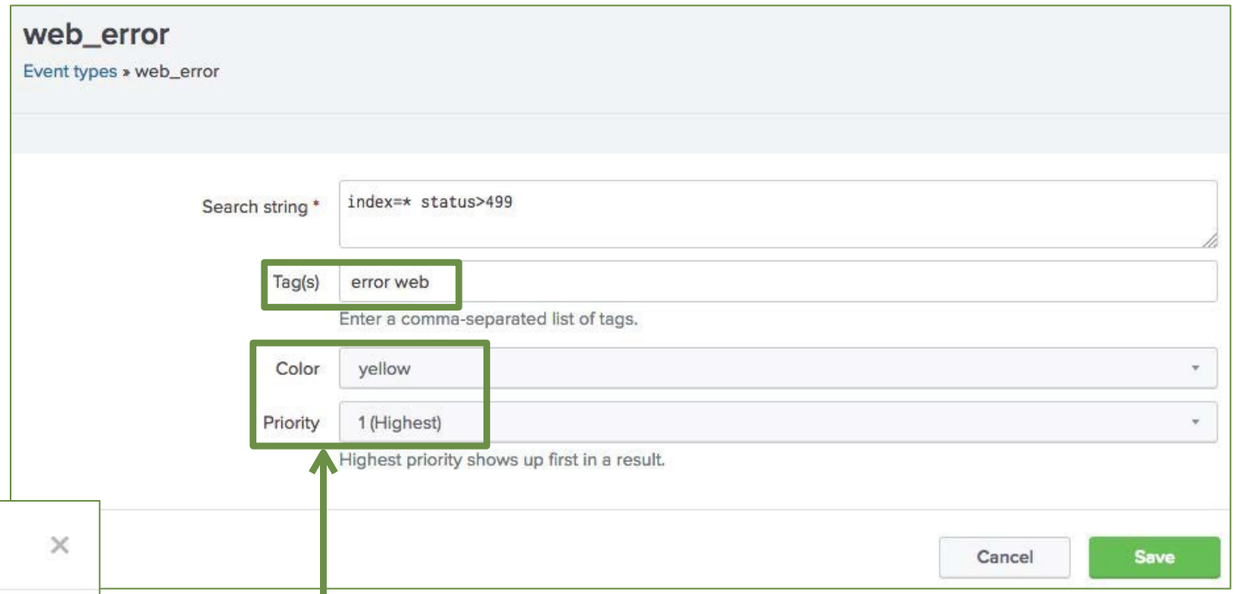
Tagging Event Types

You can tag event types two ways:

1. **Settings > Event Types**
2. **Event details > Actions**







Priority determines which event type color displays for an event

Event Types vs. Saved Reports

- Event Types
  - Categorize events based on a search string
  - Tag event types to organize data into categories
  - The `eventtype` field can be included in a search string
  - Does not include a time range
- Saved Reports
  - Search criteria will not change
  - Includes a time range and formatting of the results
  - Can be shared with Splunk users and added to dashboards

Creating and Using Macros

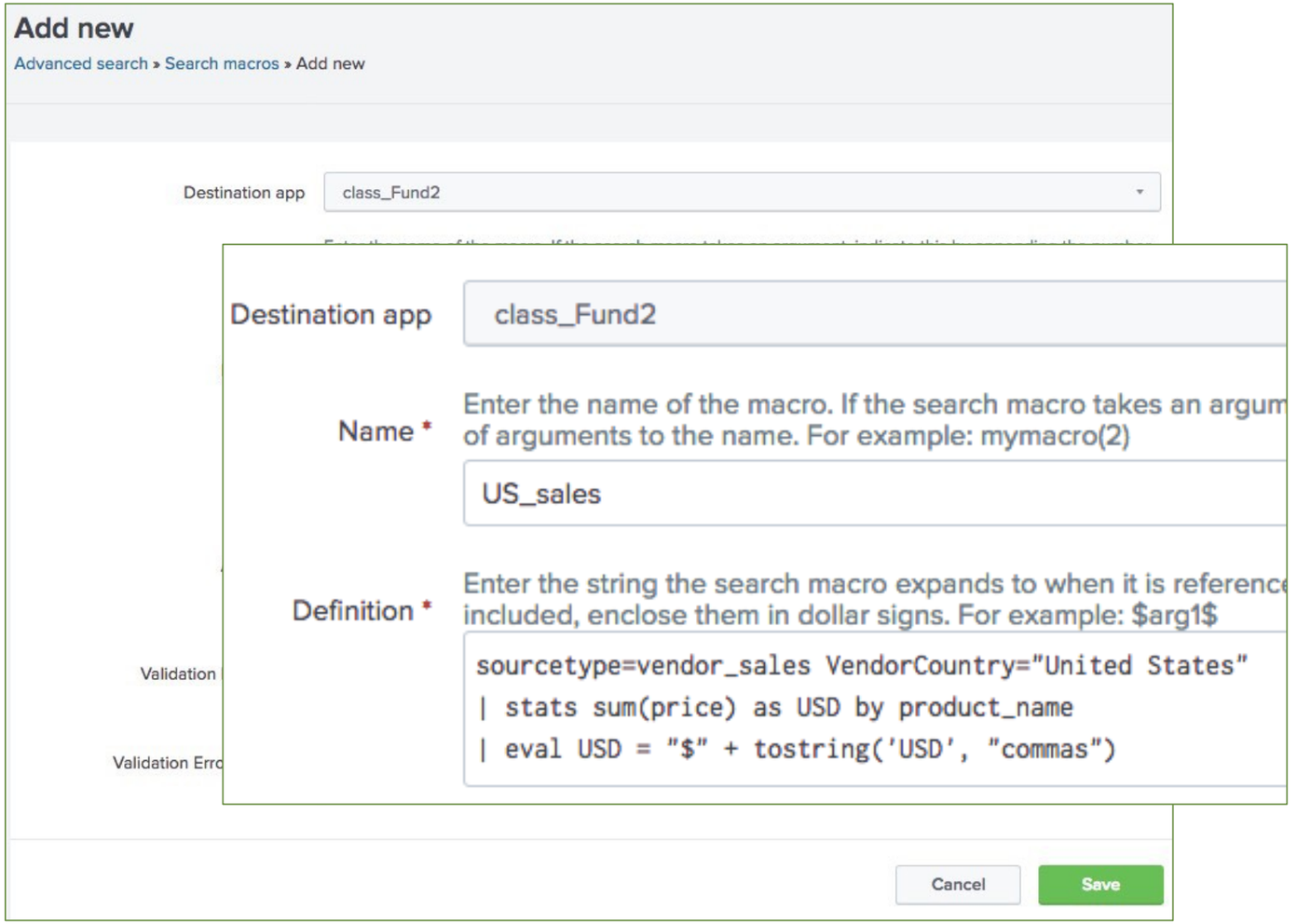
Macros Overview

- Useful when you frequently run searches or reports with similar search syntax
- The time range is selected at search time
- Macros can be a full search string or a portion of a search that can be reused in multiple places
- Allows you to define one or more arguments within search segment
  - Pass parameter values to macro at execution time
  - Macro uses values to resolve search string

Creating a Basic Macro

Settings > Advanced search > Search Macros

1. Click New Search Macro
2. Select the destination app
3. Enter a name
4. Type the search string
5. Save

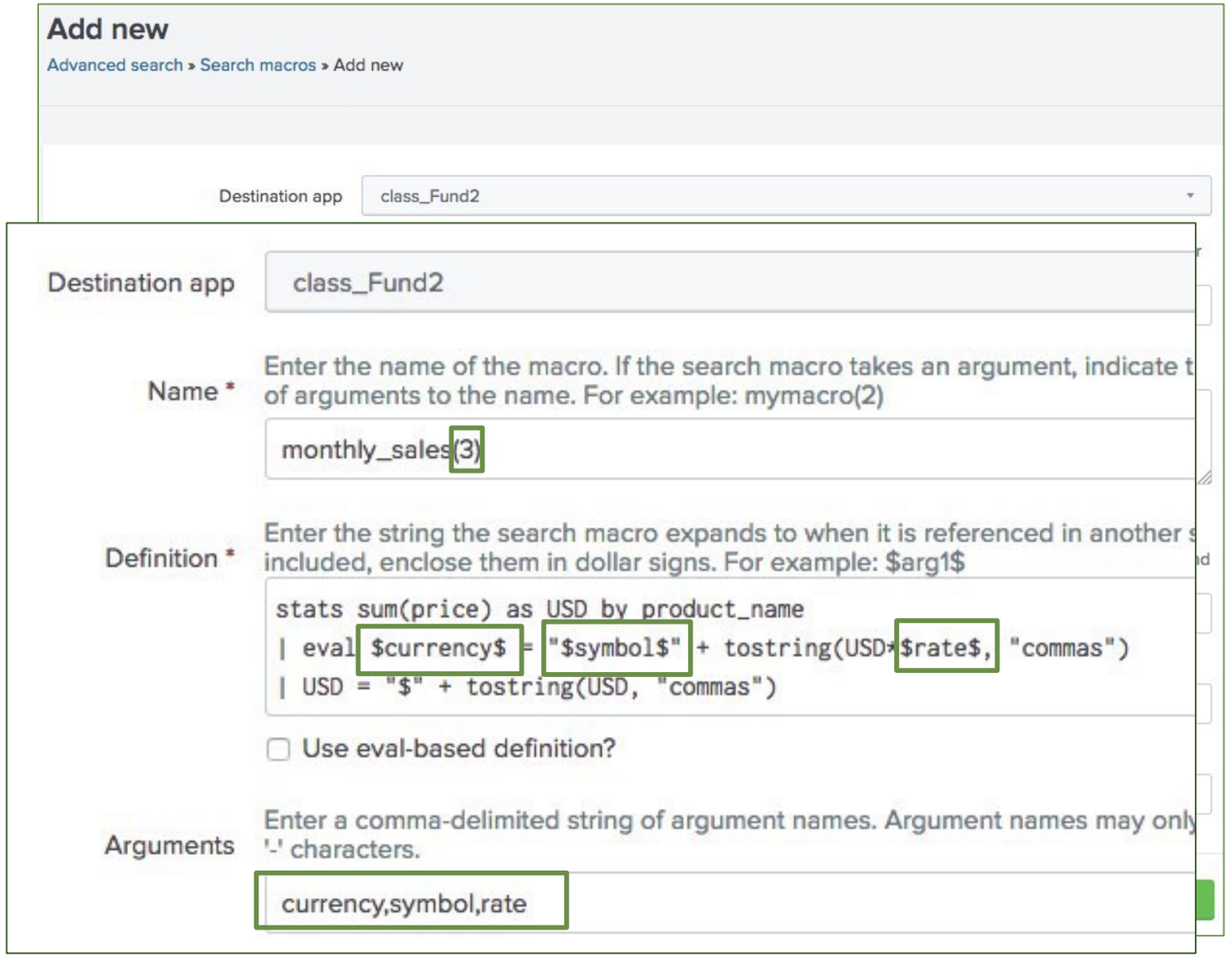


Using a Basic Macro

- Type the macro name into the search bar
- Surround the macro name with the **backtick** (or grave accent) character
  - **`macroname`** != ‘macroname’
  - Do not confuse with single-quote character (‘)
- Pipe to more commands, or precede with search string

Adding Arguments

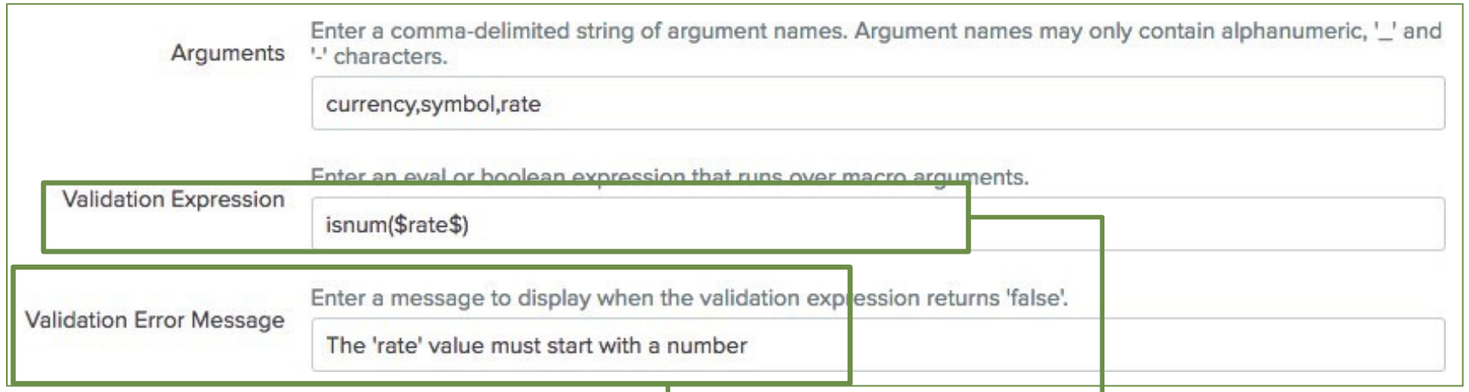
- Include the number of arguments in parentheses after the macro name
  - monthly_sales(3)
- Within the search definition, use arg
  - currency=$currency$
  - symbol=$symbol$
  - rate=$rate$
- In the **Arguments** field, enter the name of the argument(s)
- Provide one or more variables of the macro at search time

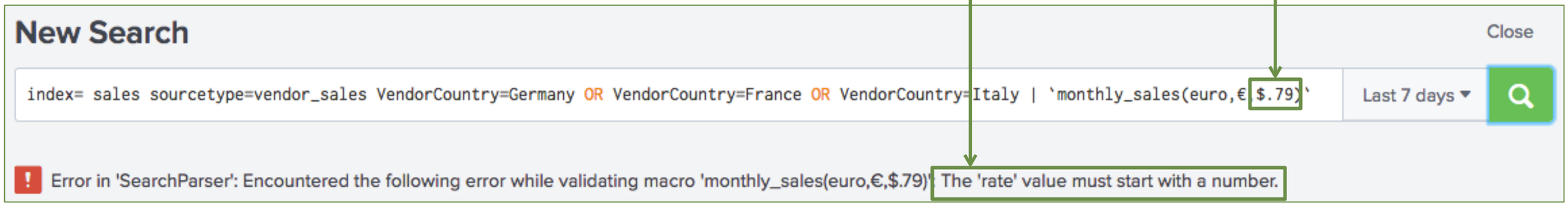


Validating Macros

You can validate argument values in your macro

- Validation Expression: you can enter an expression for each argument
  - Argument must be enclosed in dollar signs
- Validation Error Message: message that appears when you run the macro

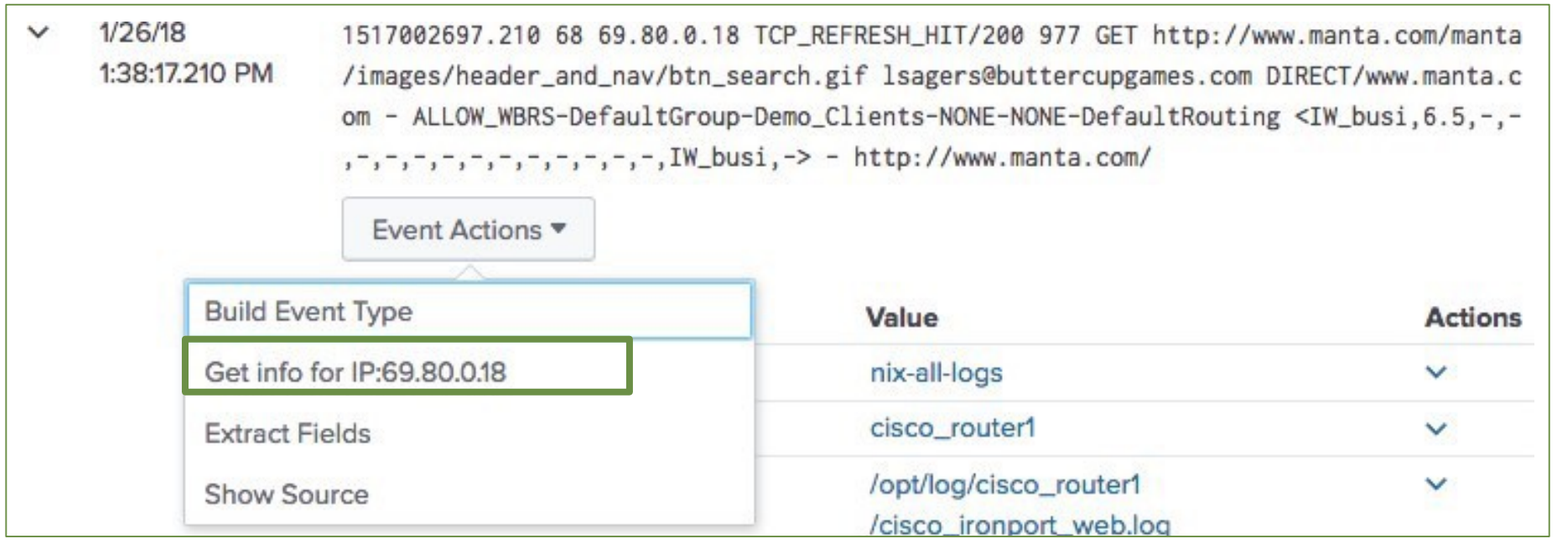




Creating and Using WorkflowActions

What are Workflow Actions?

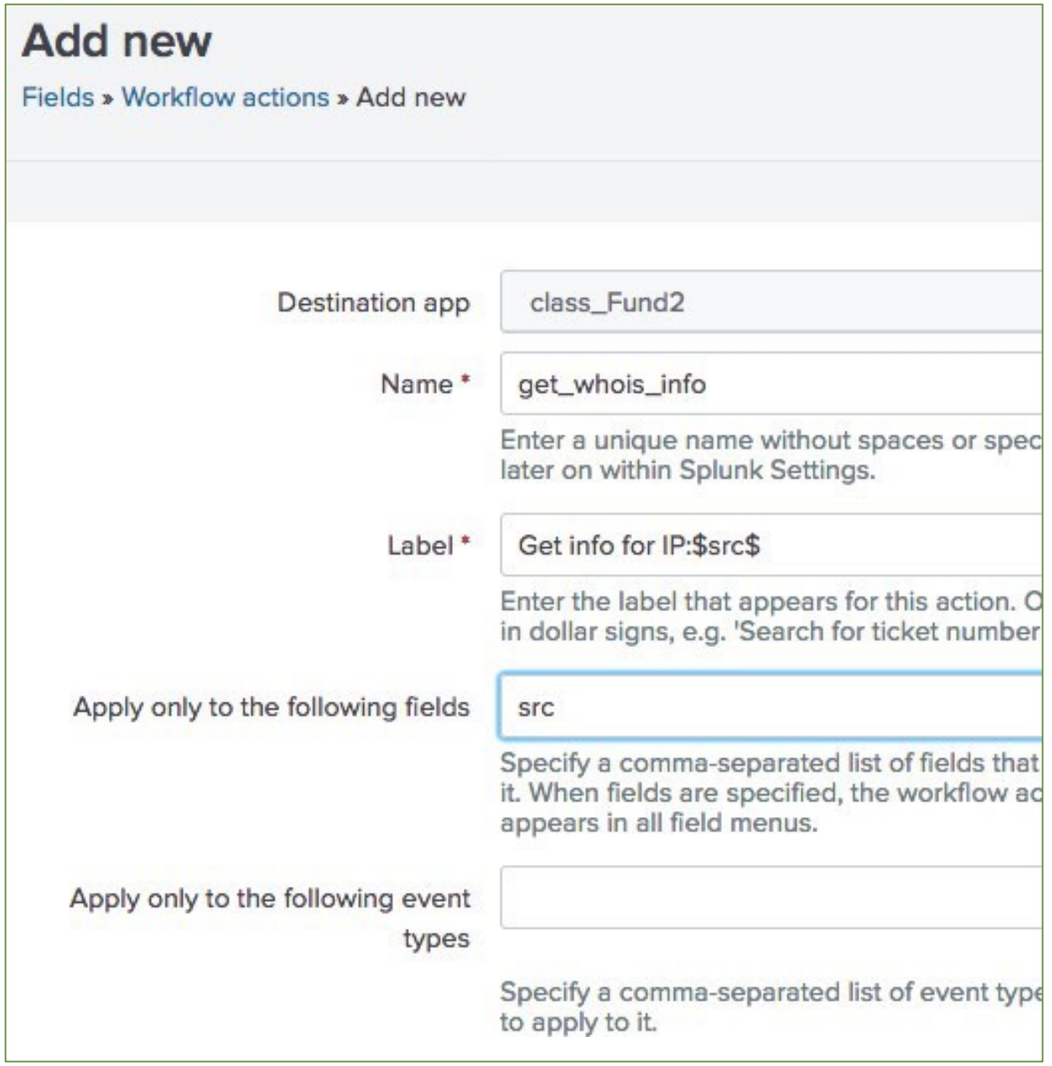
- Execute workflow actions from an event in your search results to interact with external resources or run another search
  - **GET** - retrieve information from an external resource
  - **POST** - send field values to an external resource
  - **Search** - use field values to perform a secondary search



Creating a GET Workflow Action

Settings > Fields > Workflow actions > New Workflow Action

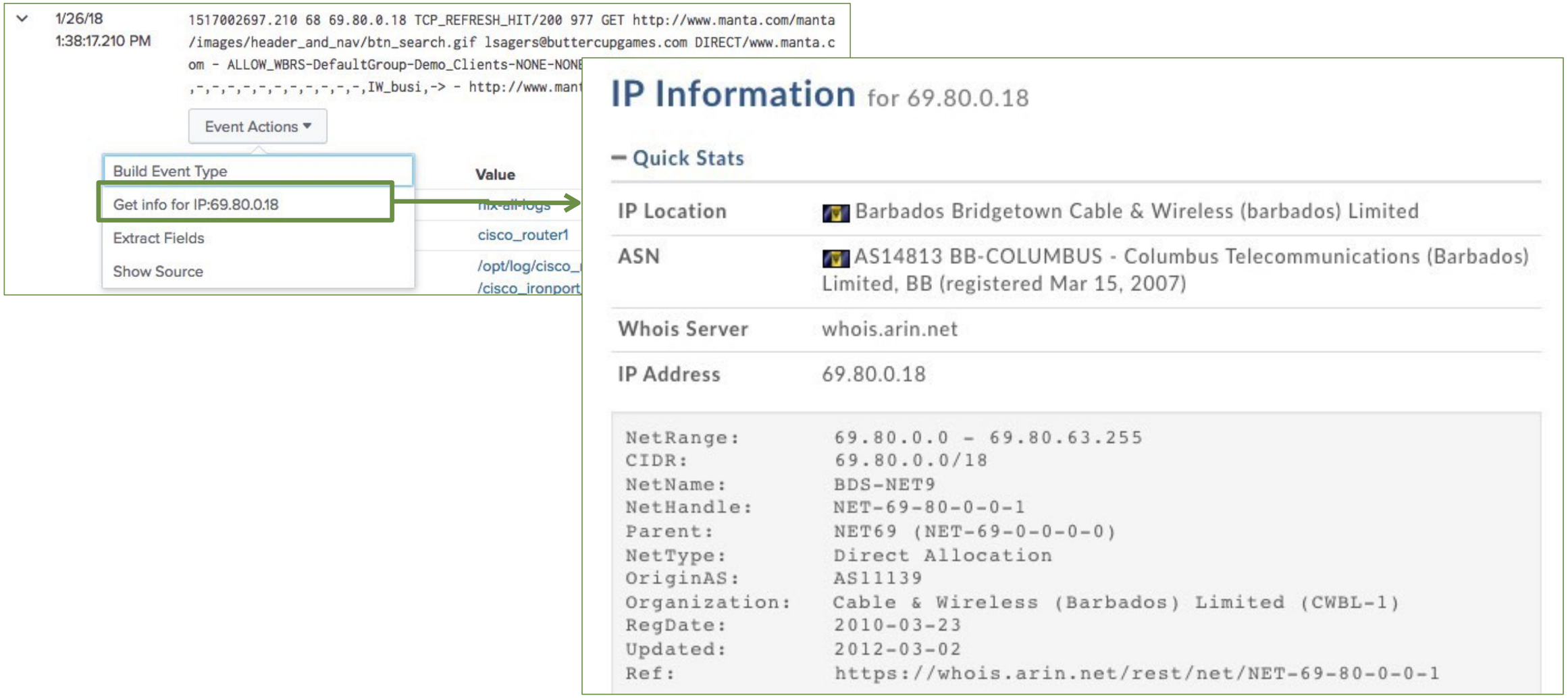
1. Select the app
2. Name the workflow action with no spaces or special characters
3. Define the label, which will appear in the Event Action menu
4. Determine if your workflow action applies to a field or event type



Creating a GET Workflow Action (cont.)

5. From the **Show action in** drop-down list, select **Event menu**
6. From Action type dropdown list, select link
7. Enter the URI of where the user will be directed
8. Specify if the link should open in a New window or Current window
9. Select the Link method of **get**
10. Save

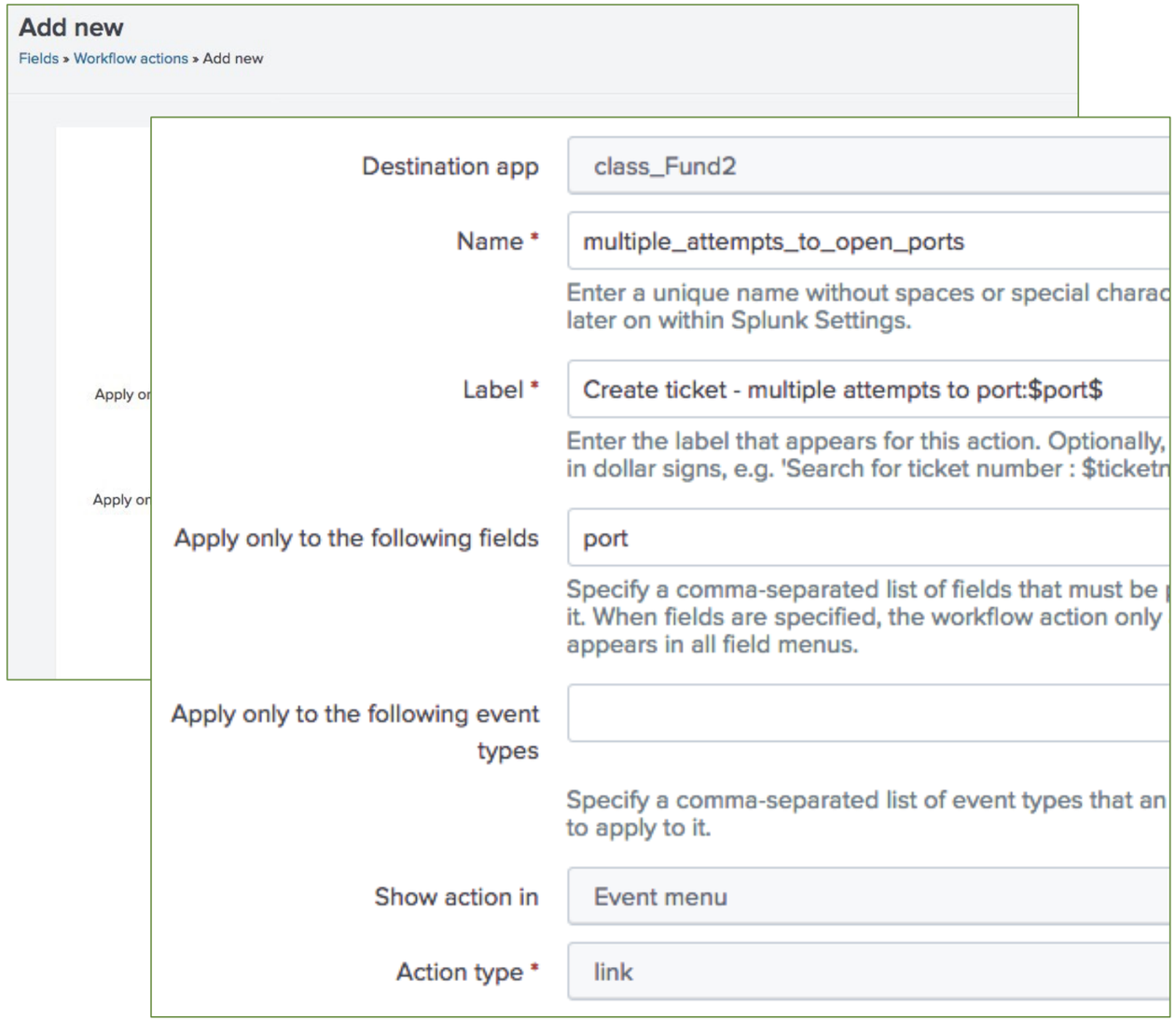
Testing the GET Workflow Action



Creating a POST Workflow Action

Settings > Fields > Workflow actions > New Workflow Action

Complete steps 1 – 6 as described in the previous example, Creating a GET Workflow Action



Creating a POST Workflow Action (cont.)

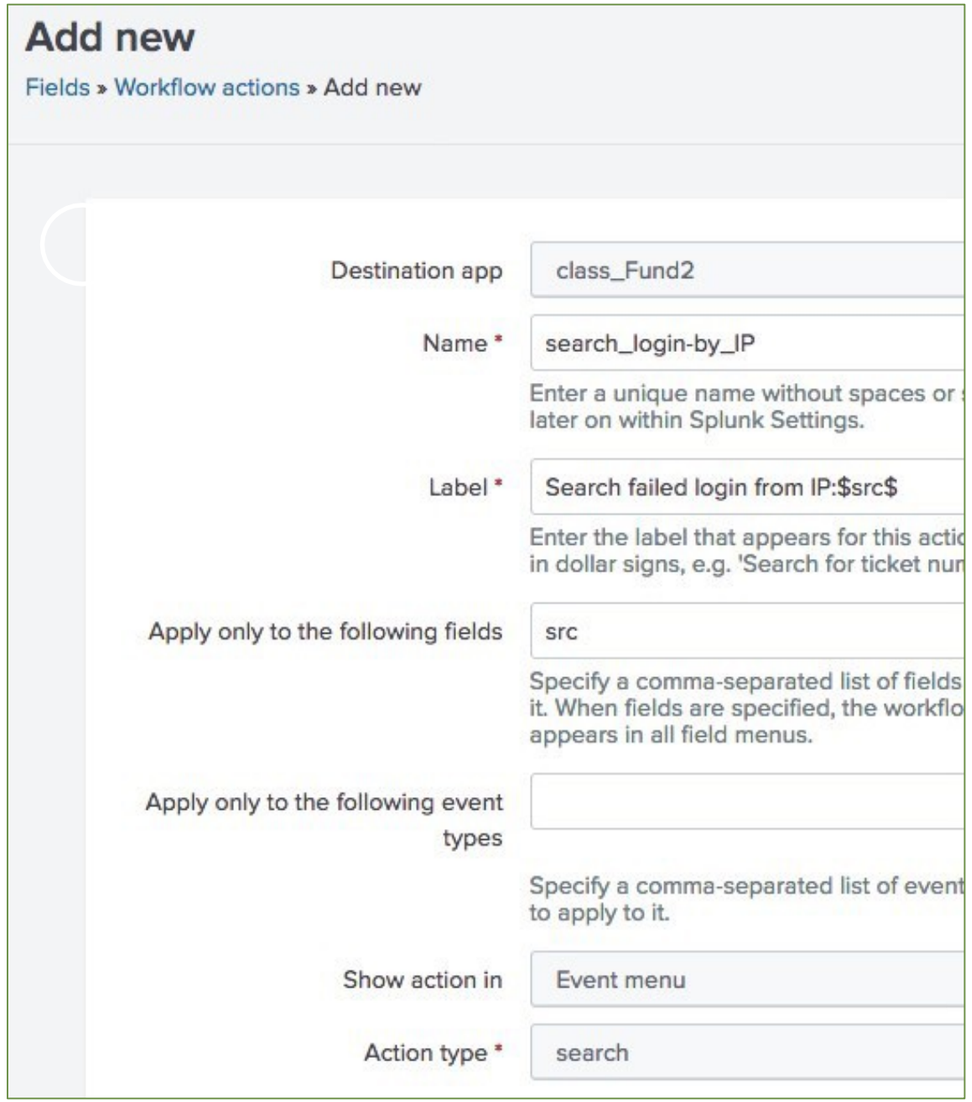
7. Enter the URI of where the user will be directed
8. Open the link in a **New window** or Current window
9. Select the Link method of **post**
10. Provide post argument parameters
11. Save

Creating a Search Workflow Action

Settings > Fields > Workflow actions> New

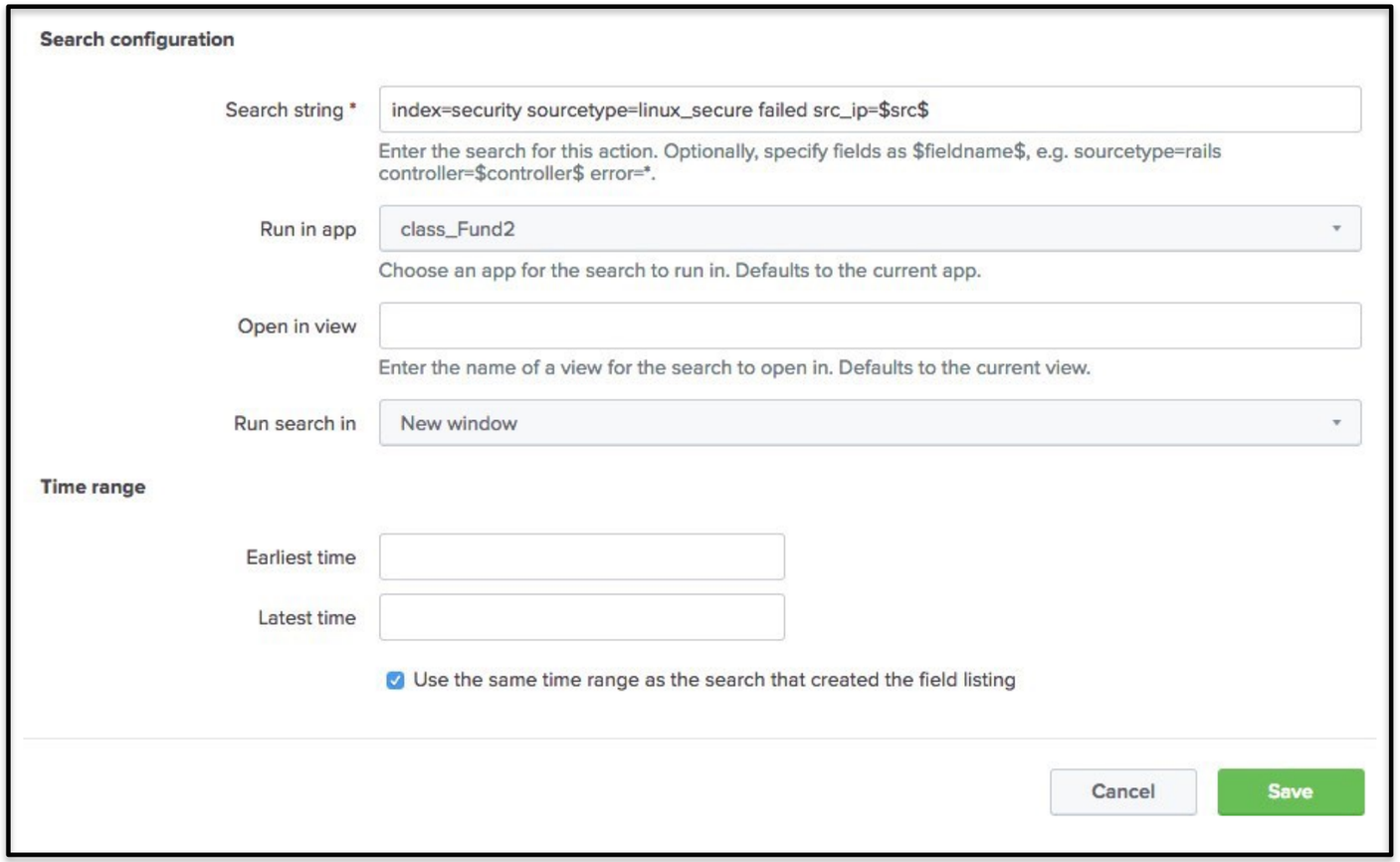
Complete steps 1 – 5 as described in the previous example, Creating a GET Workflow Action

6. From the Action type dropdown list, select **search**

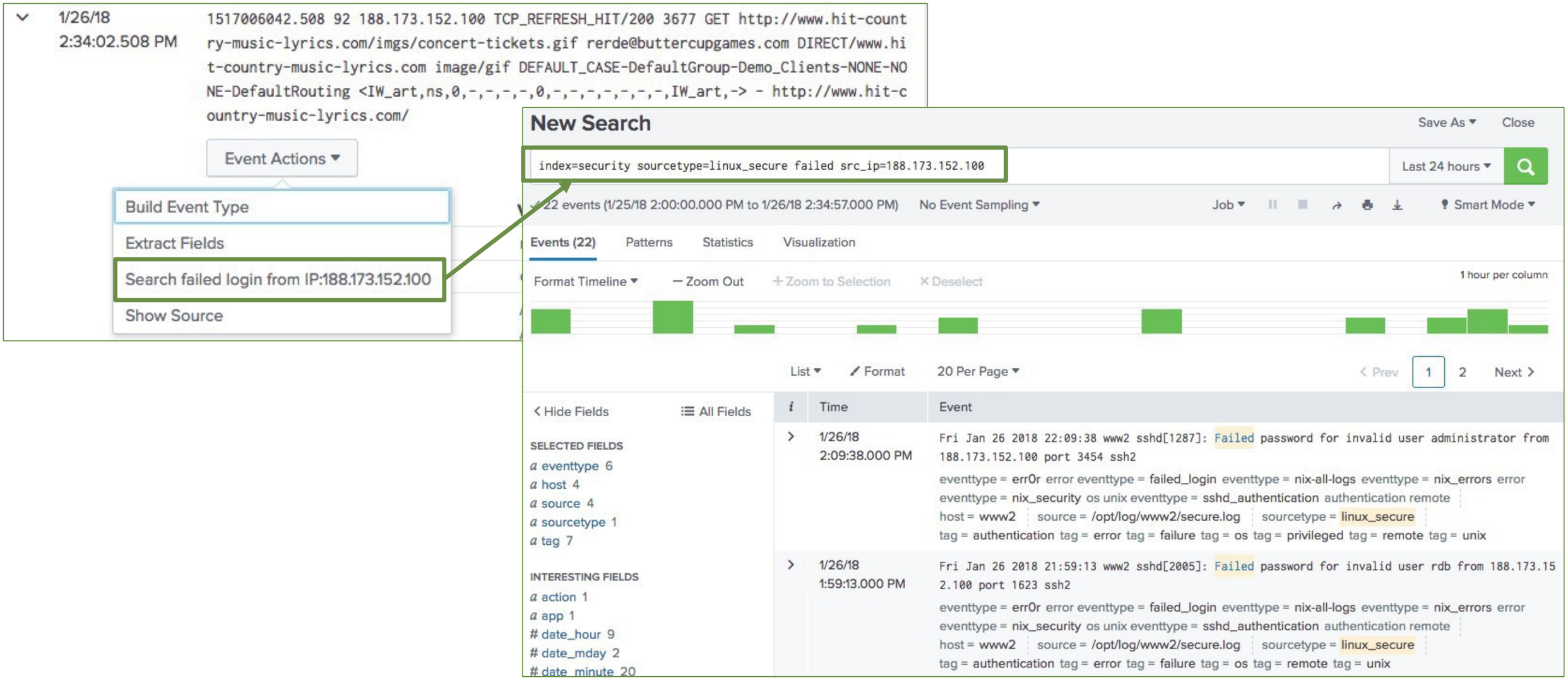


Creating a Search Workflow Action (cont.)

7. Enter the Search string
8. Select the app if it is different from the current app
9. Enter the view name where the search will execute
10. Indicate if the search should run in a New window or the Current window
11. Enter the time range for the search or choose to use the same time range as the search
12. Save



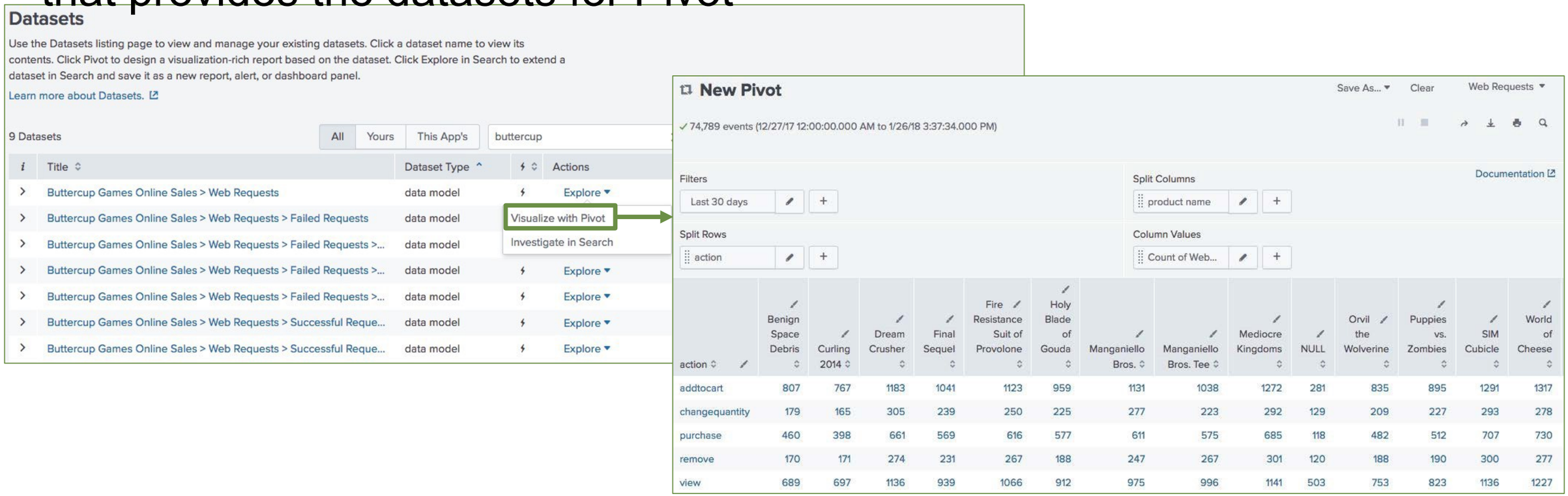
Testing the Search Workflow Action



Creating Data Models

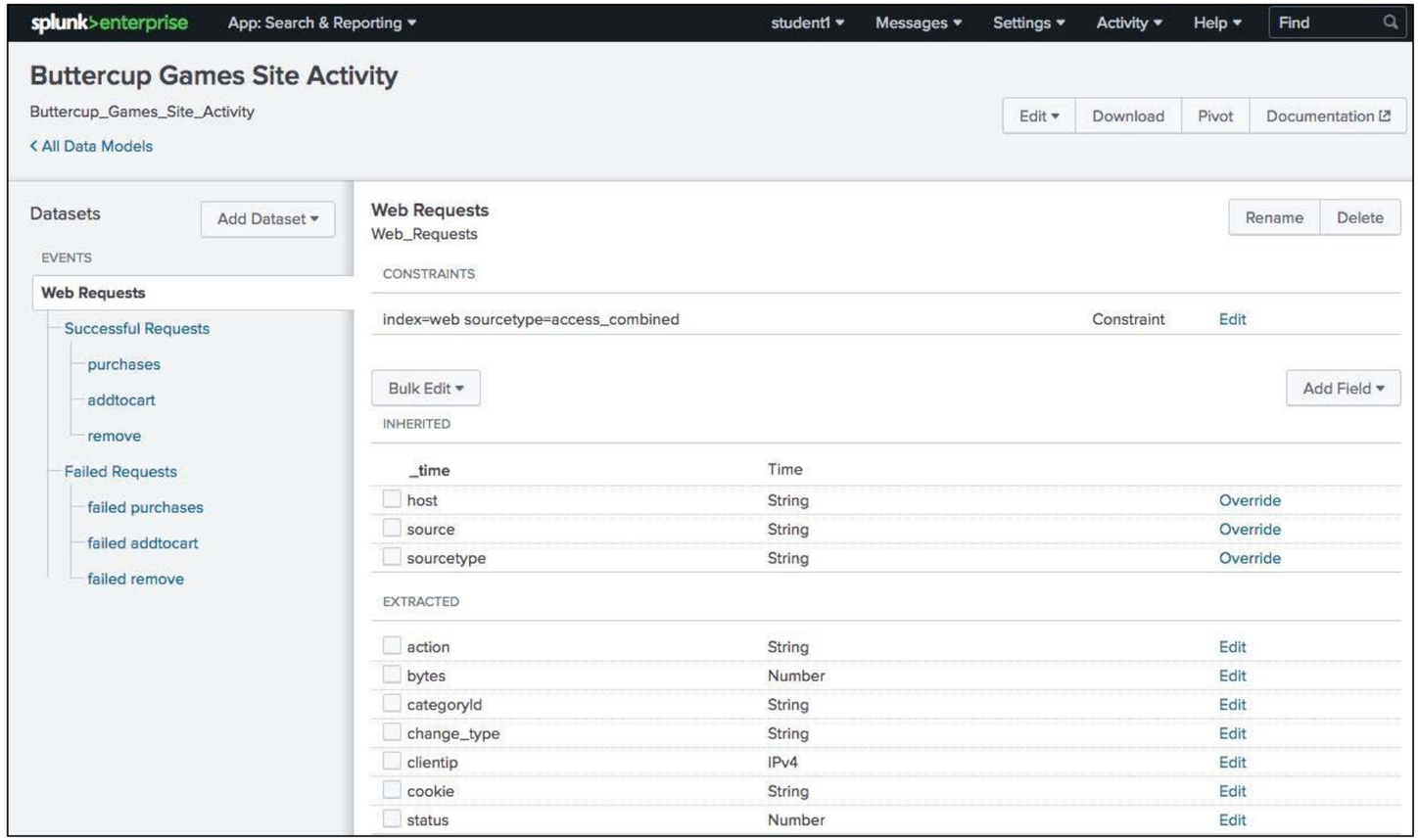
Reviewing Pivot

- Used for creating reports and dashboards (discussed in Splunk Fundamentals 1)
- As a knowledge manager, you're responsible for building the data model that provides the datasets for Pivot



Overview of Data Models

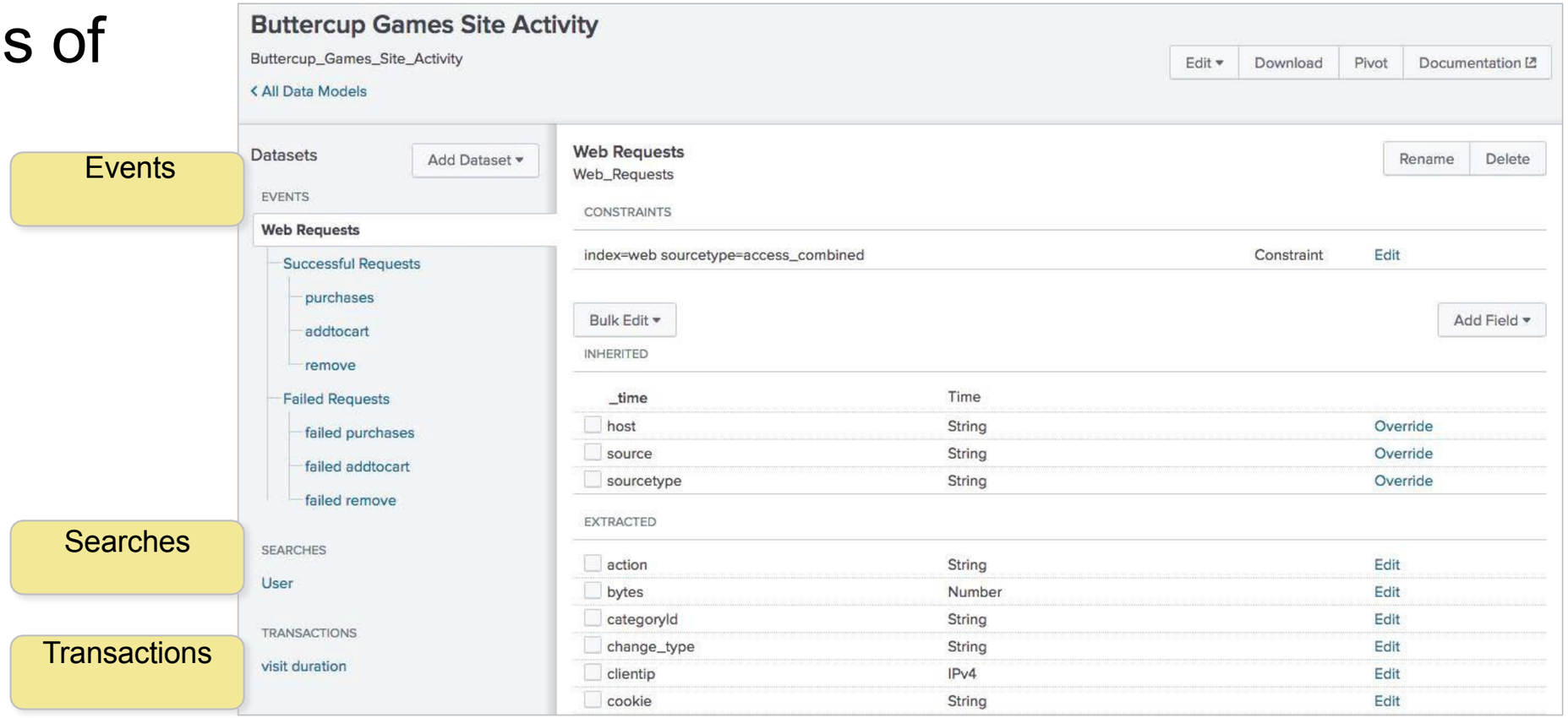
- Hierarchically structured datasets that generate searches and drive Pivot
- Pivot reports are created based on datasets
- Each event, search or transaction is saved as a separate dataset



Data Model Dataset Types

A data model can consist of 3 types of datasets

- Events
- Searches
- Transactions



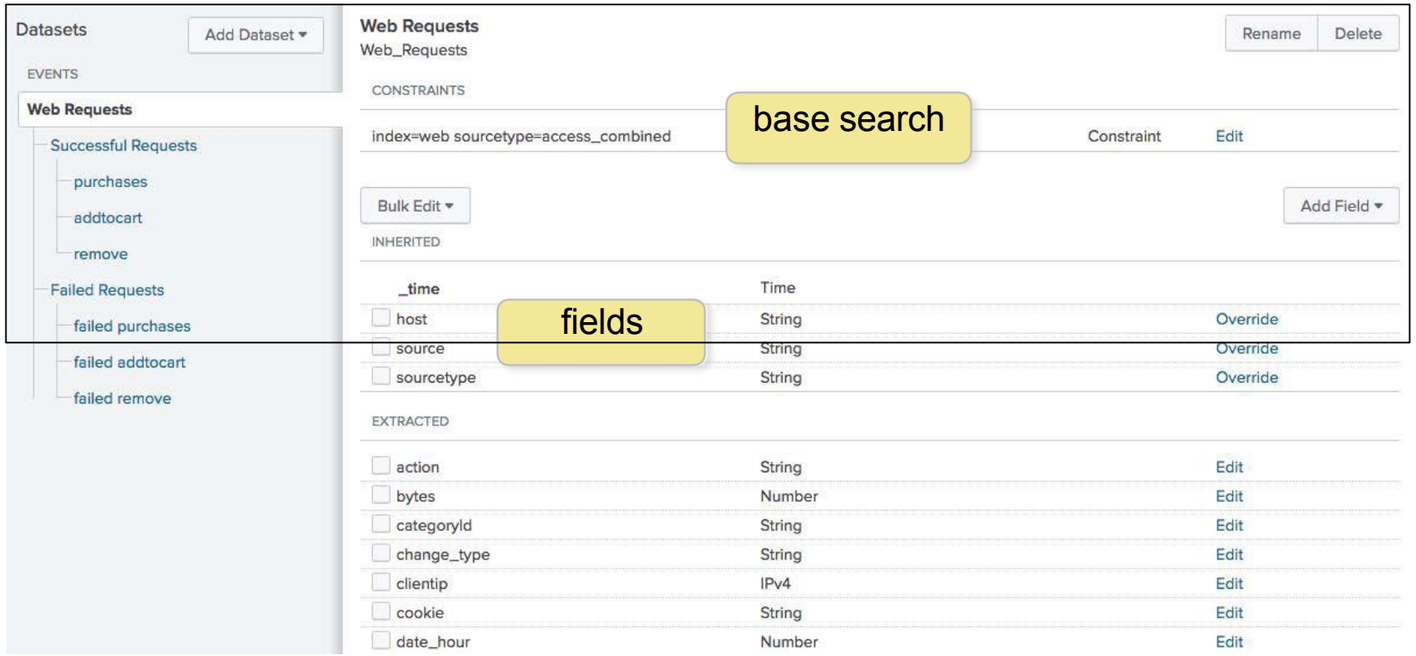
Events

Searches

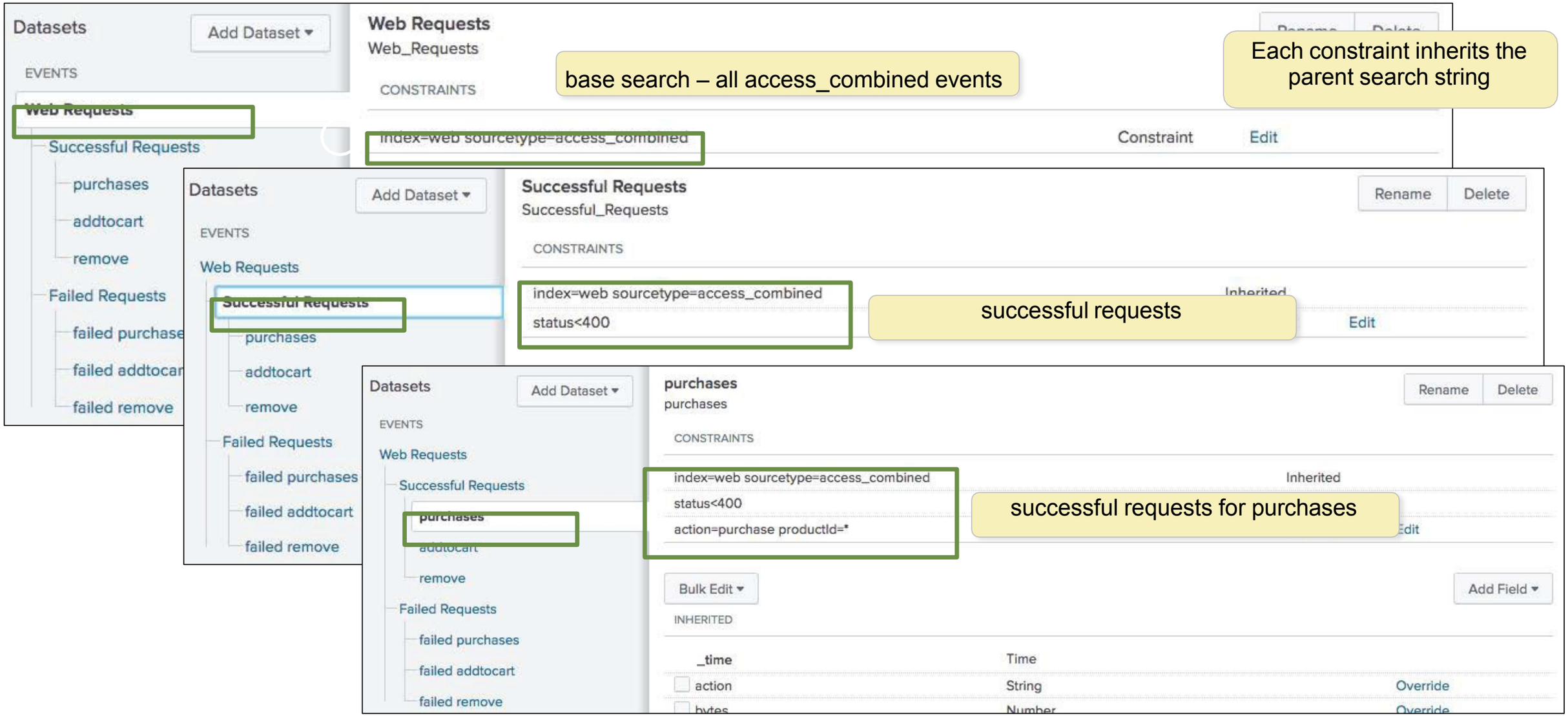
Transactions

Data Model Events

- **Event datasets** contain constraints and fields
- **Constraints** are essentially the search broken down into a hierarchy
- **Fields** are properties associated with the events

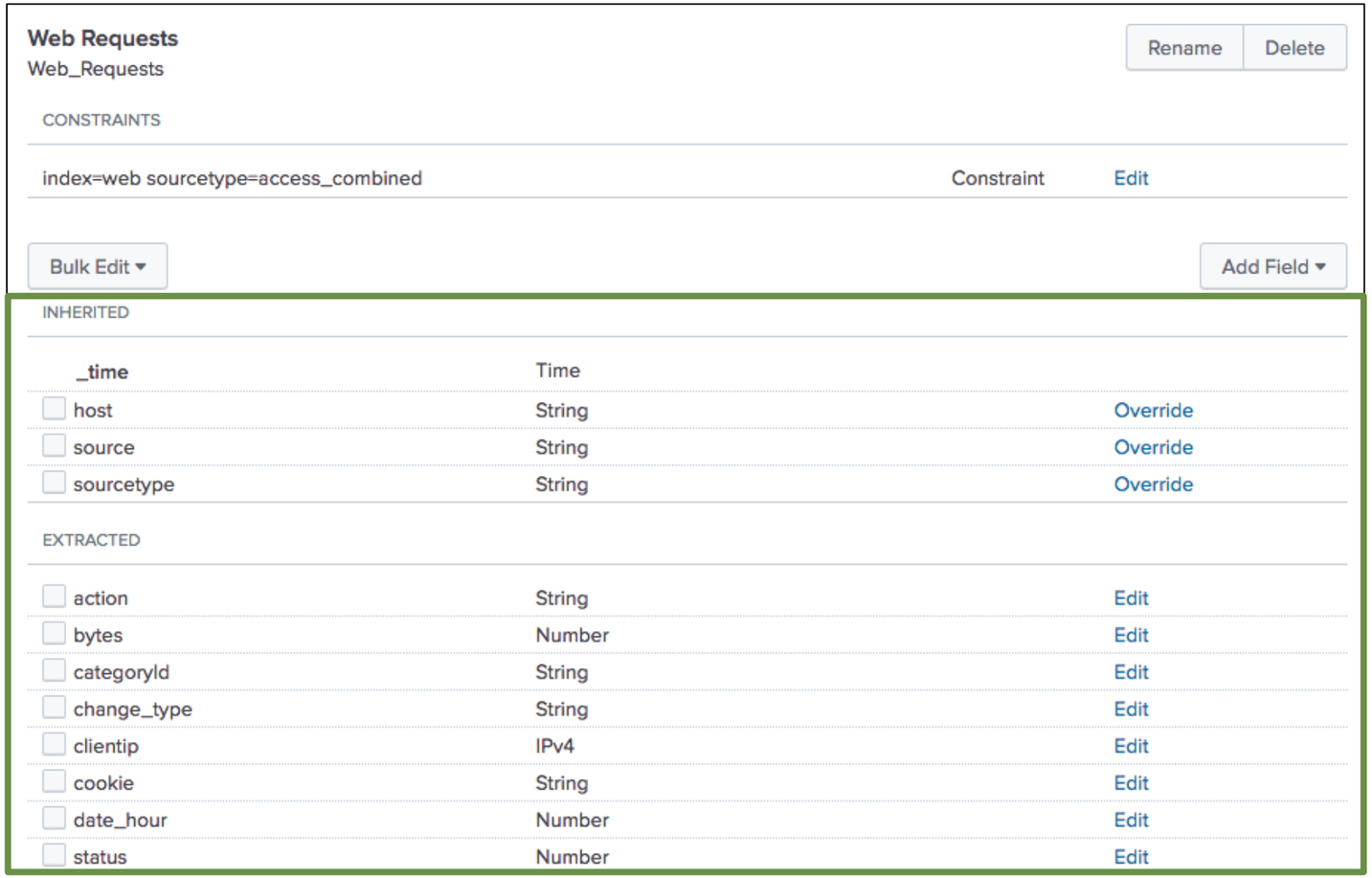


Event Object Hierarchy and Constraints



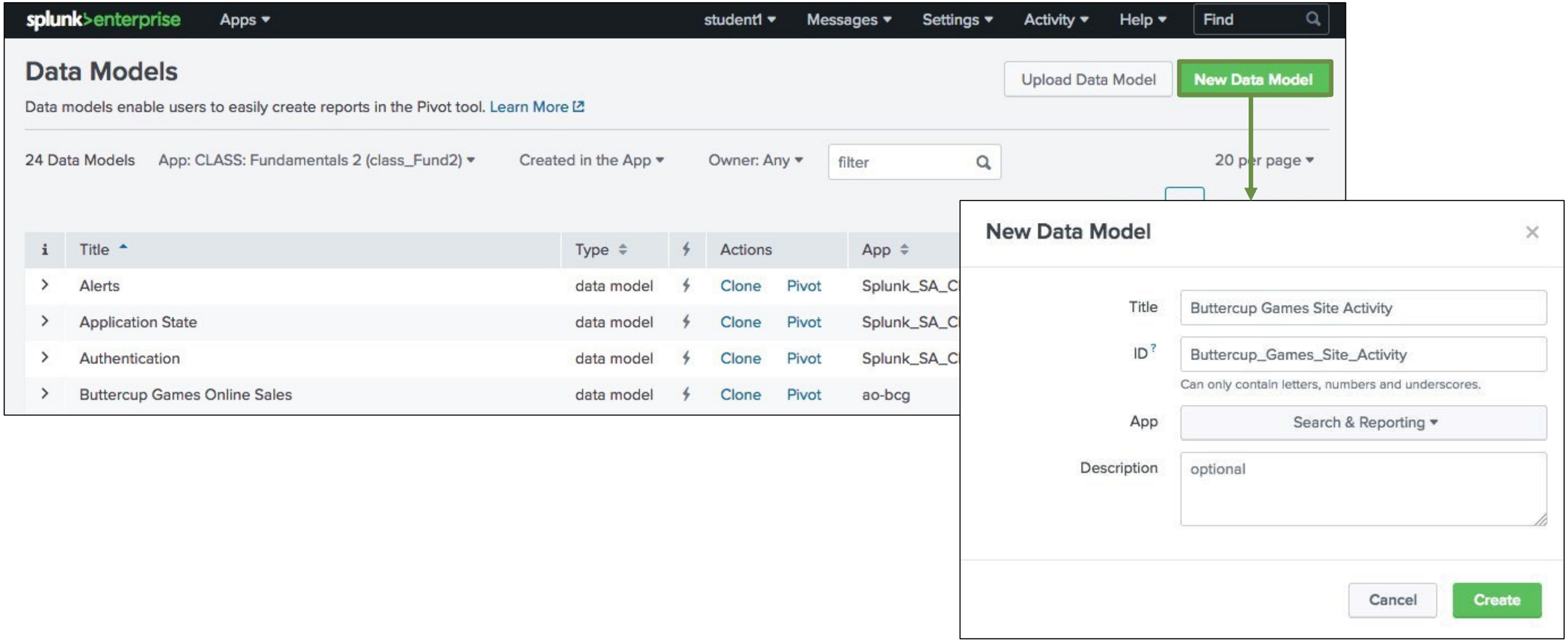
Dataset Fields

- Select the fields you want to include in the dataset
- Like constraints, fields are inherited from parent objects

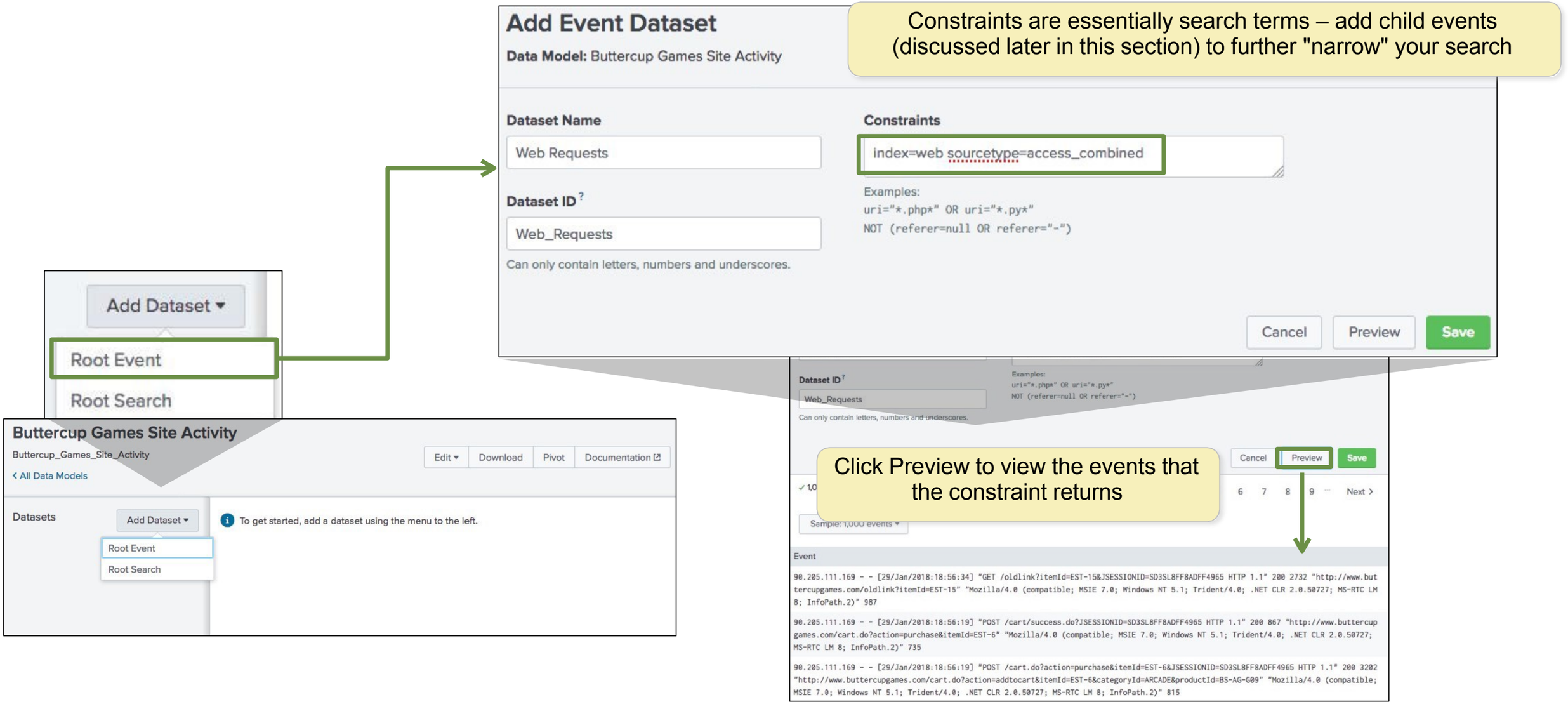


Creating a Data Model

Settings > Data Models

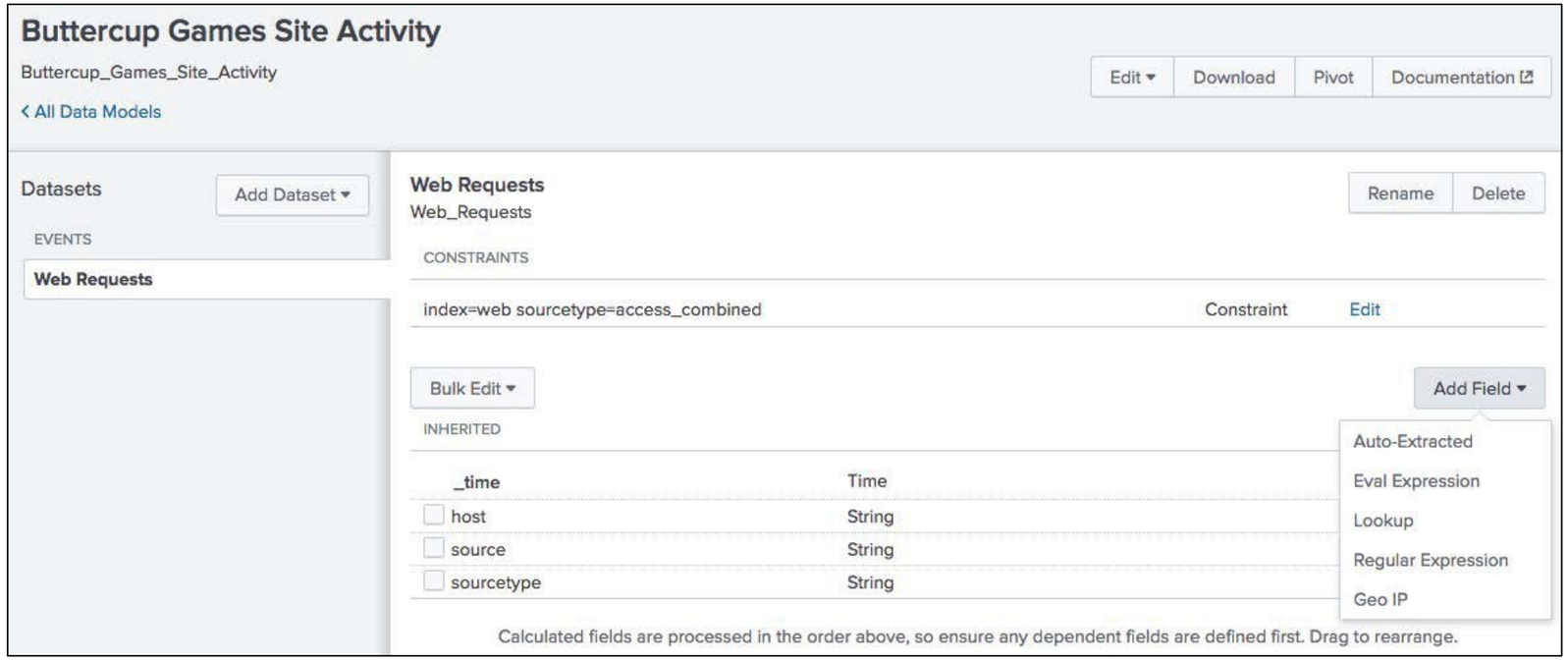


Adding a Root Event



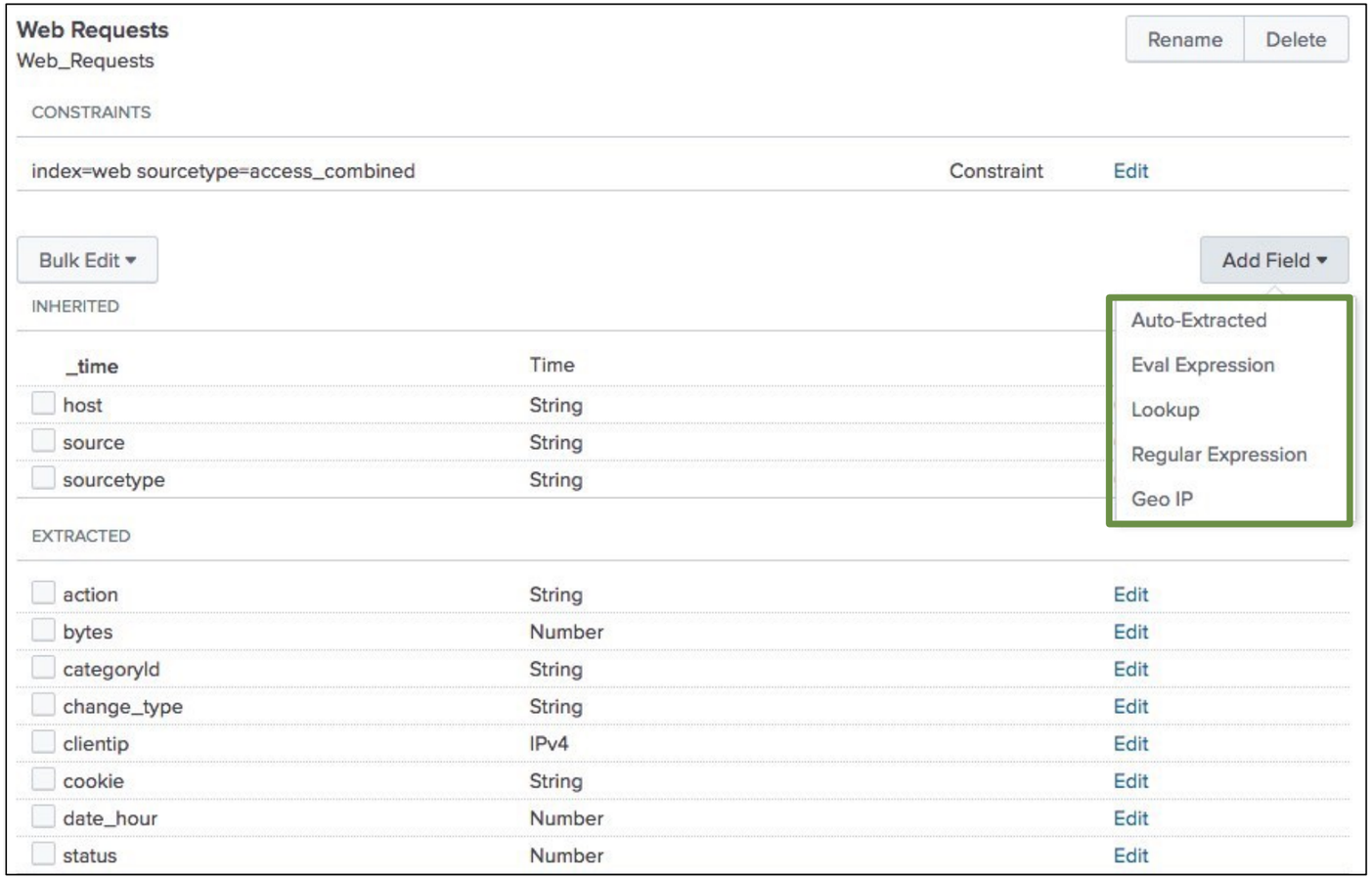
Adding a Root Event (cont.)

- In this example, the root event of this data model represents all web requests
- The Inherited attributes are default fields
- Use **Add Field > Auto-Extracted** to add more fields



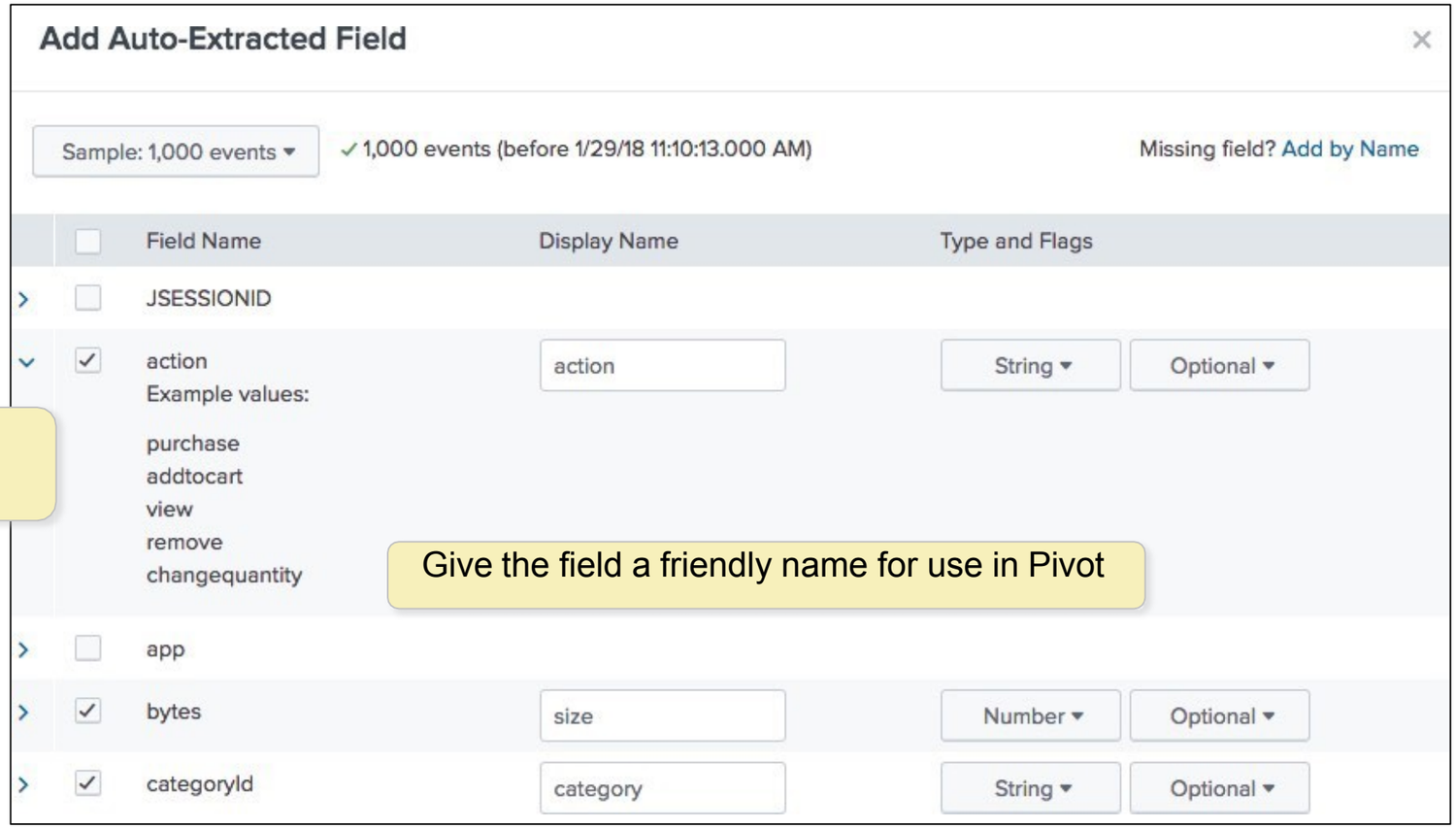
Adding Fields

- **Auto-Extracted** – can be default fields or manually extracted fields
- **Eval Expression** – a new field based on an expression that you define
- **Lookup** – leverage an existing lookup table
- **Regular Expression** – extract a new field based on regex
- **Geo IP** – add geographical fields such as latitude/ longitude, country, etc.



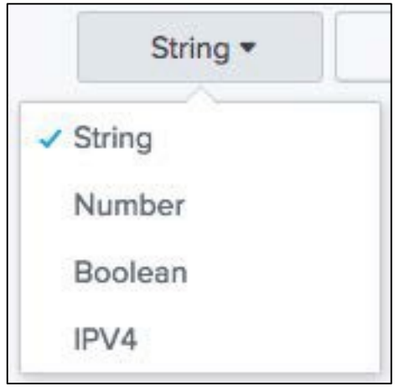
Adding Fields – Auto-Extracted

Fields that already exist for the constraint can be added as attributes to the data model



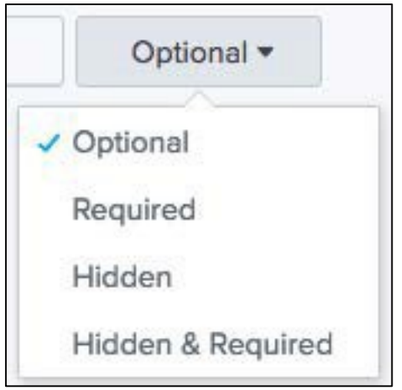
Field Types

- String: Field values are recognized as alpha-numeric
- Number: Field values are recognized as numeric
- Boolean: Field values are recognized as true/false or 1/0
- IPV4: Field values are recognized as IP addresses
  - This is an important field type, as at least one IPV4 attribute type must be present in the data model in order to add a Geo IP attribute



Field Flags

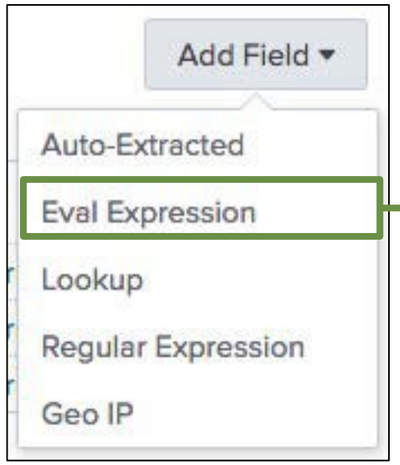
- Optional: This field doesn't have to appear in every event
- Required: Only events that contain this field are returned in Pivot

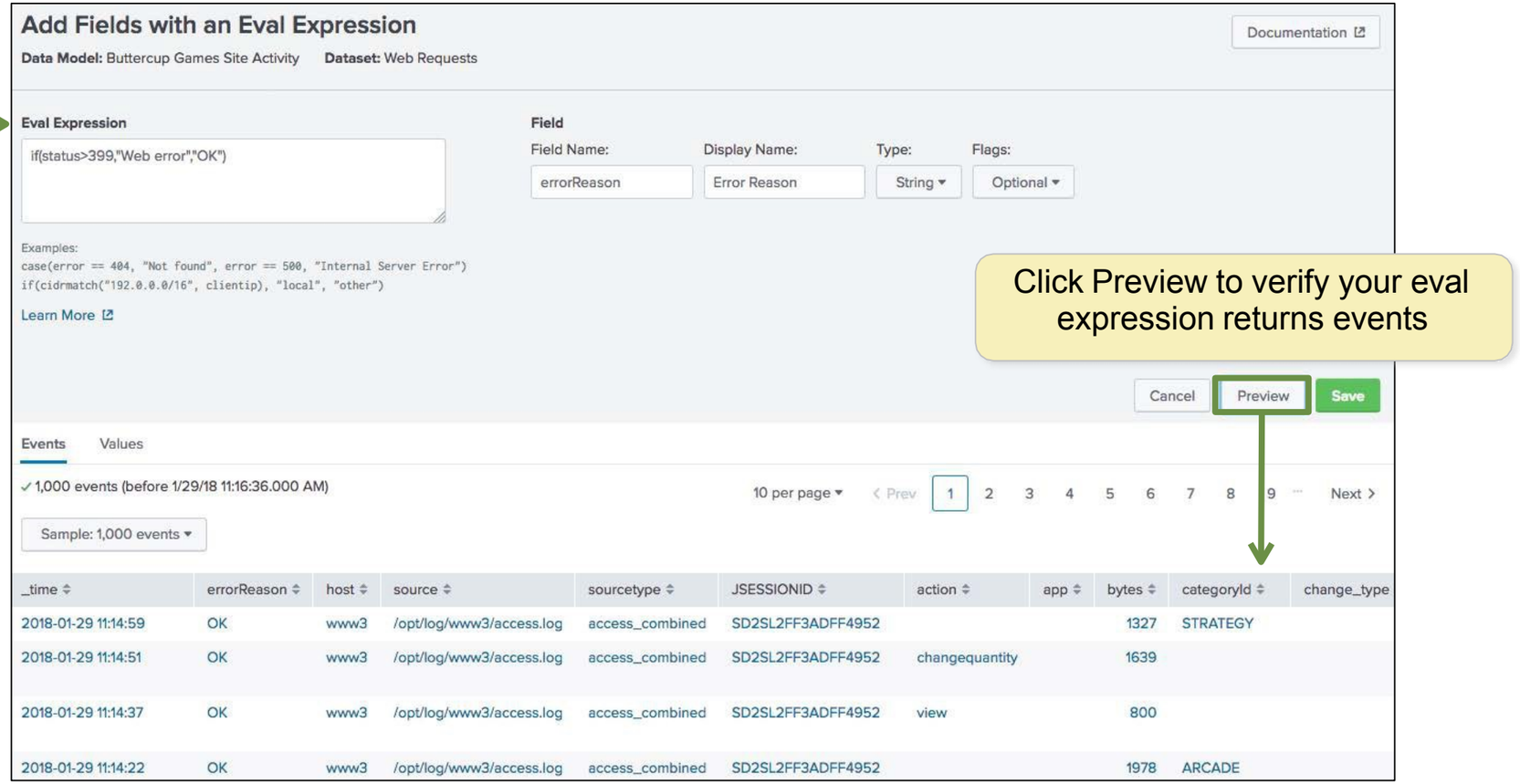


- Hidden: This field is not displayed to Pivot users when they select the dataset in Pivot
  - Use for fields that are only being used to define another field, such as an eval expression
- Hidden & Required: Only events that contain this field are returned, and the fields are hidden from use in Pivot

Adding Fields – Eval Expressions

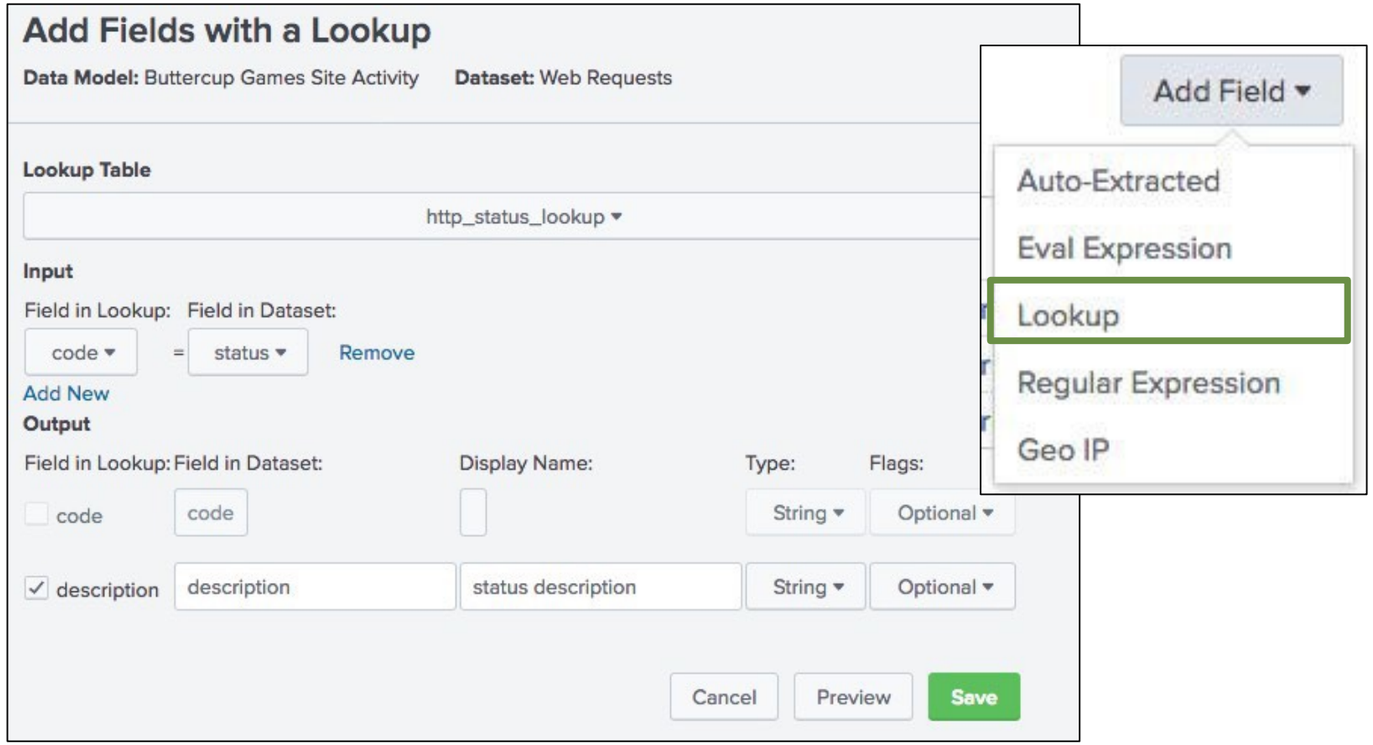
- You can define a new field using an eval expression
  - In this example, you create a field named Error Reason that evaluates the value of the status field





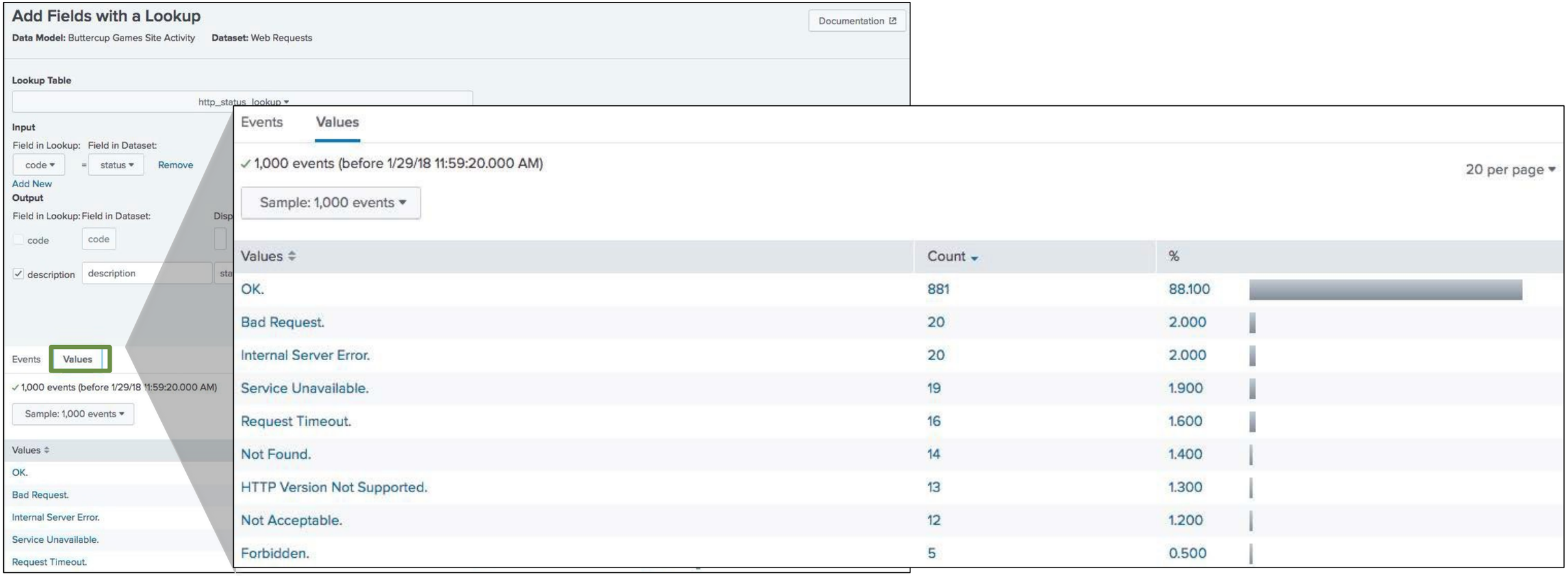
Adding Fields – Lookups

- Leverage an existing lookup definition to add fields to your event object
- Configure the lookup attribute in the same way as an automatic lookup



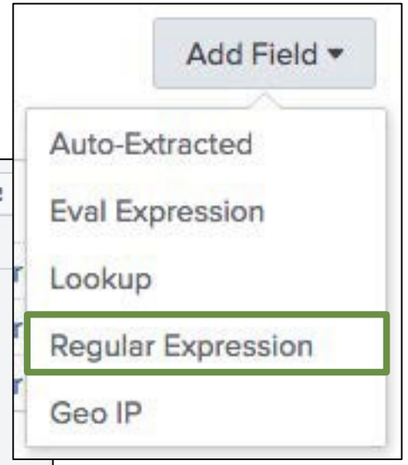
Adding Fields – Lookups (cont.)

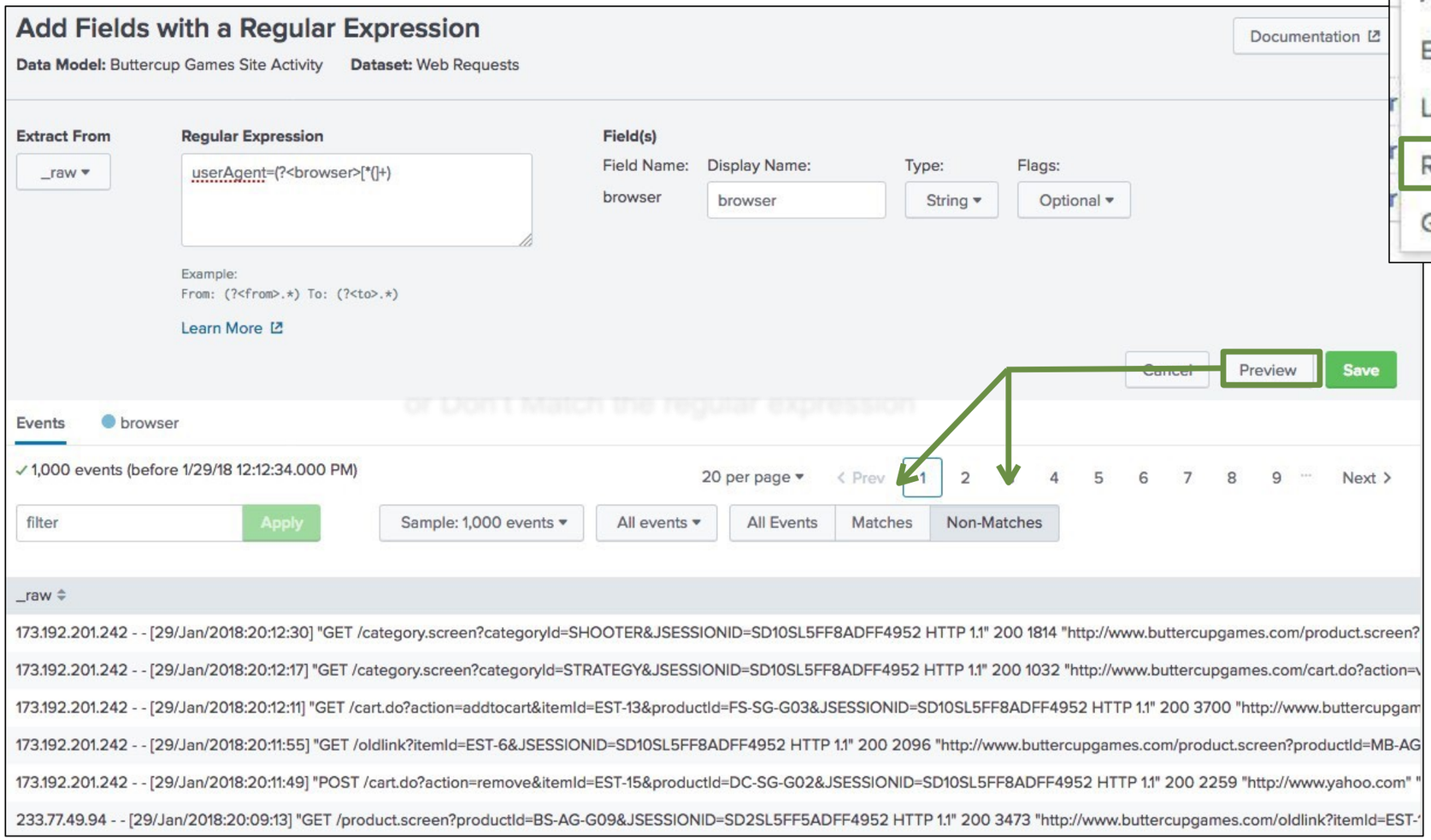
- Use Preview to test your lookup settings
- Use the Events and Values tab to verify your results



Adding Fields – Regular Expression

You can define a new field using a regular expression



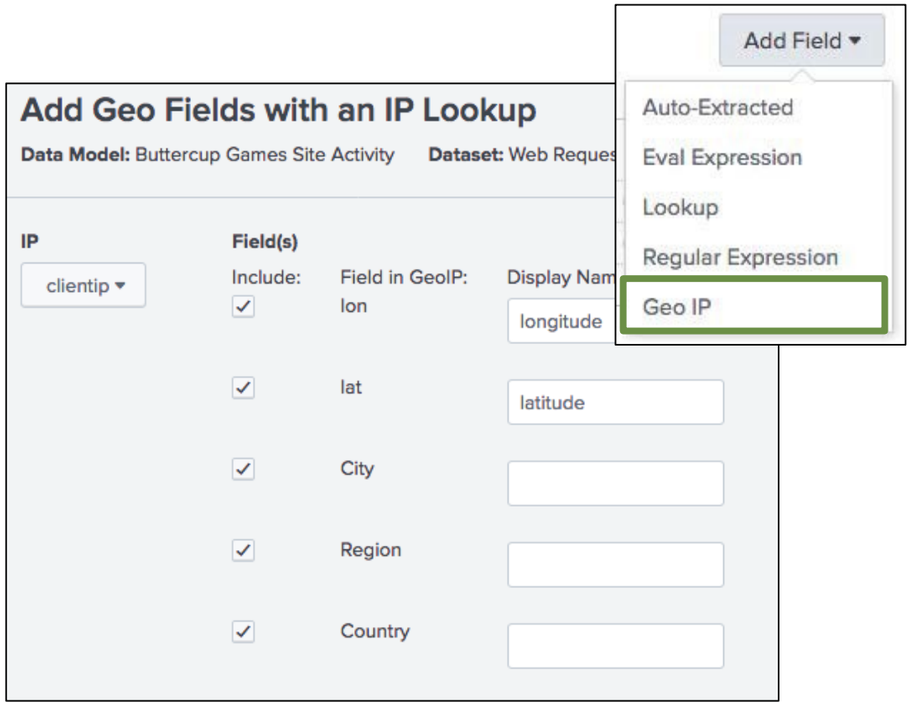


Adding Fields - GeoIP

- Map visualizations require latitude/longitude fields
- To use Geo IP Lookup, at least one IP field must be configured as an IPv4 type
- While the map function isn't available in Pivot, the data model can be called using the

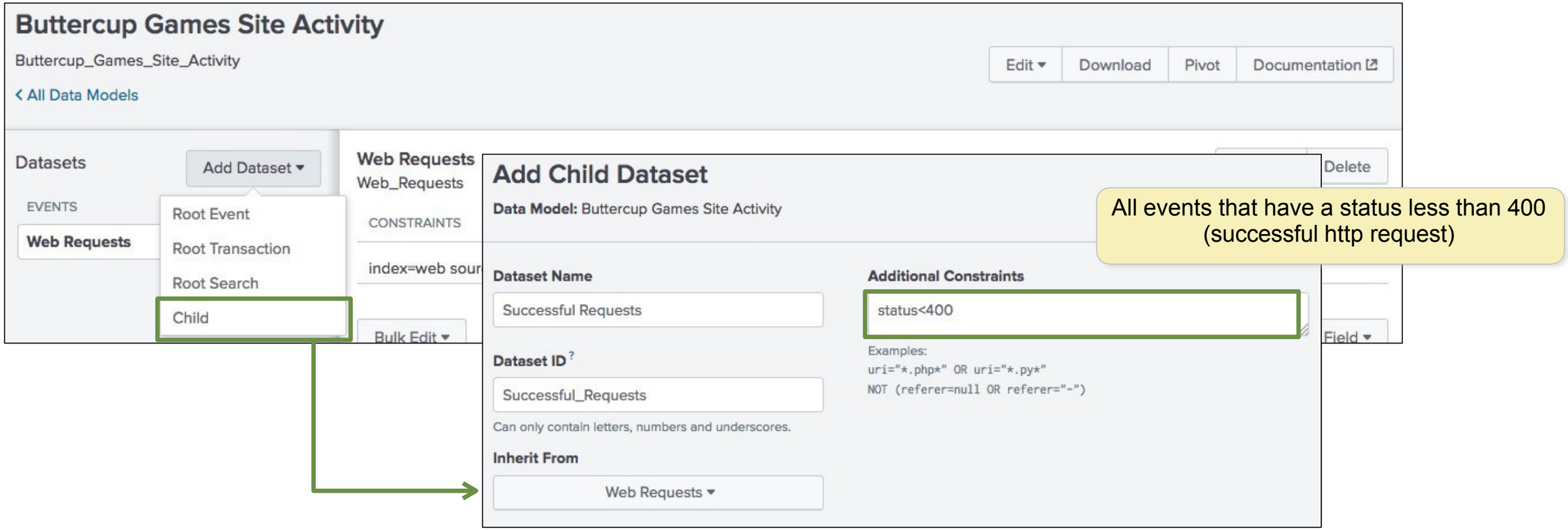
  | pivot command and <map>

  element in a dashboard population search
  - Select the field that contains the mapping to lat/lon
  - Identify the lat/lon and geo fields in the data



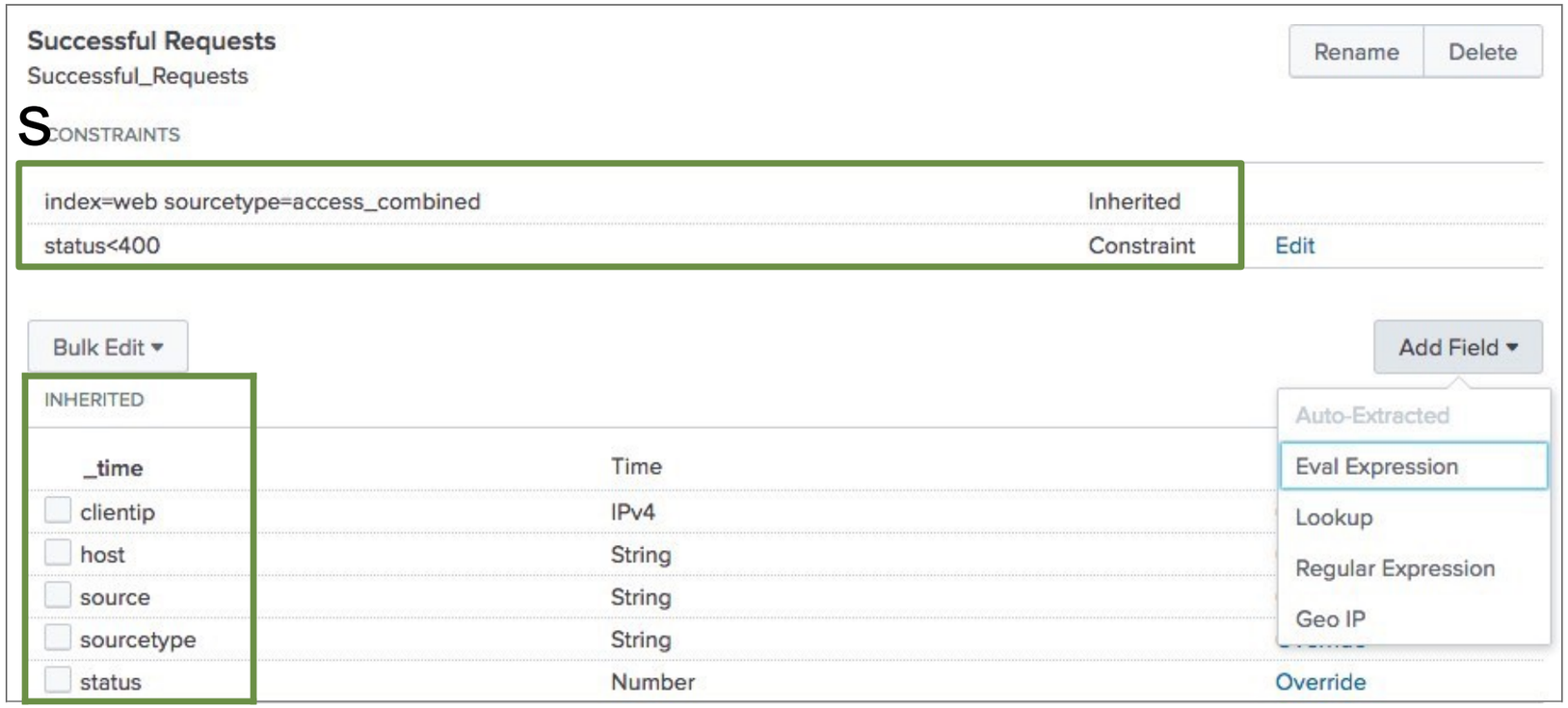
Adding Child Datasets

When you create a new child dataset, you give it one or more additional constraints



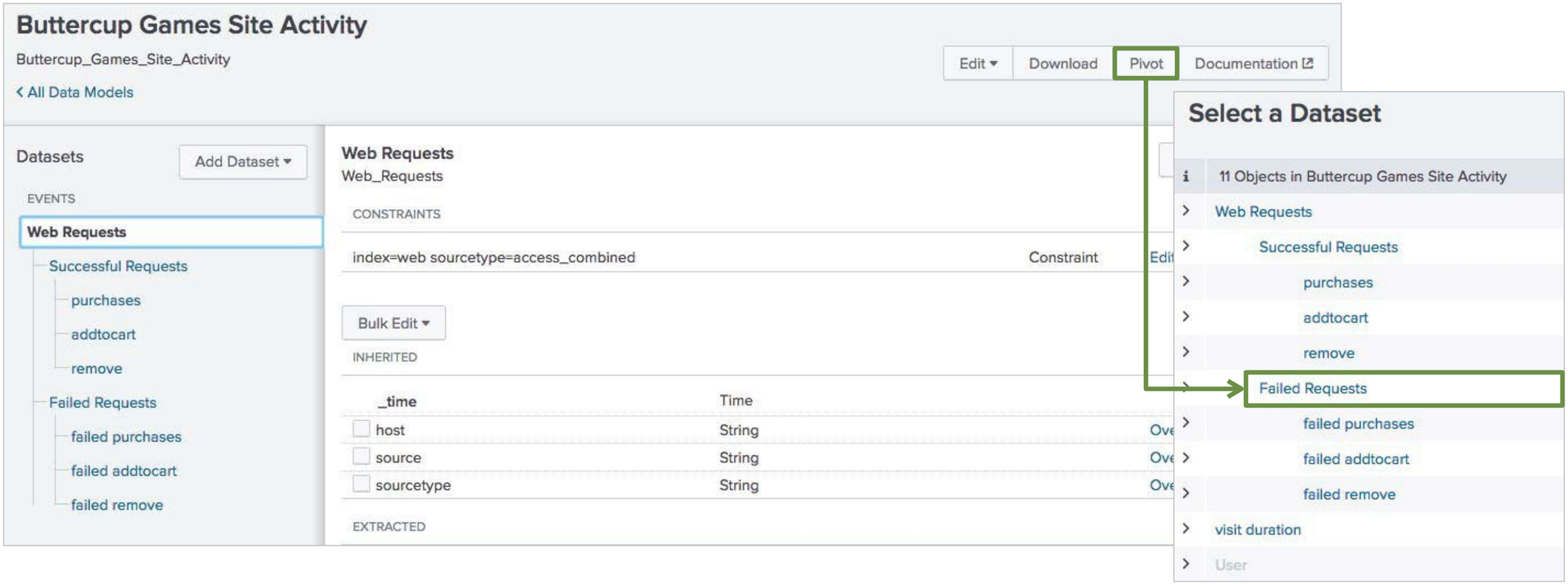
Adding Child Datasets (cont.)

- Child datasets inherit all fields from the parent events
- You can add more fields to child datasets



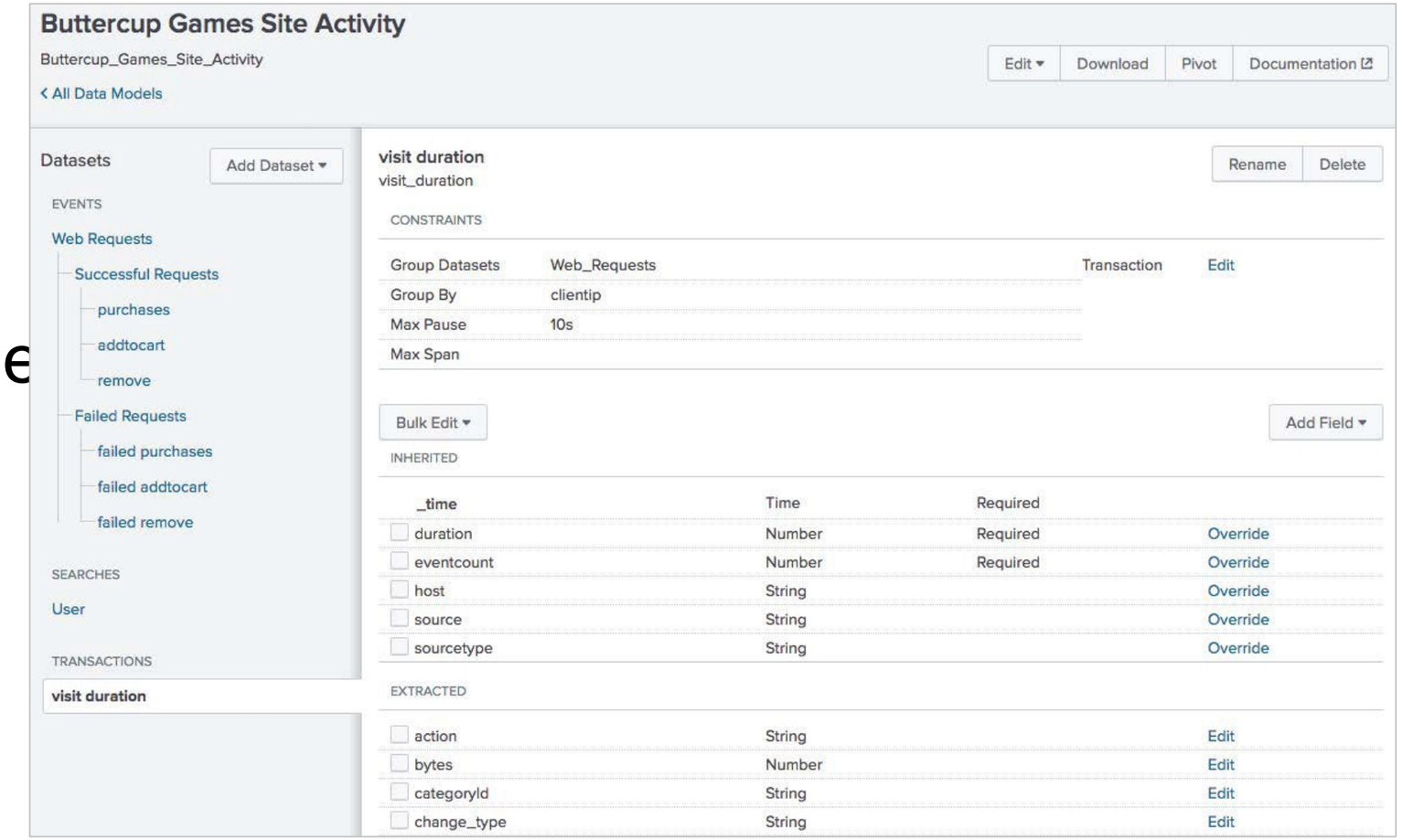
Testing the Data Model

- Click Pivot to access the Select a Dataset window
- Choose an object from the selected data model to begin building the report



Data Model Transaction Datasets

- Enable the creation of datasets that represent transactions
- Use fields that have already been added to the model using event or search datasets

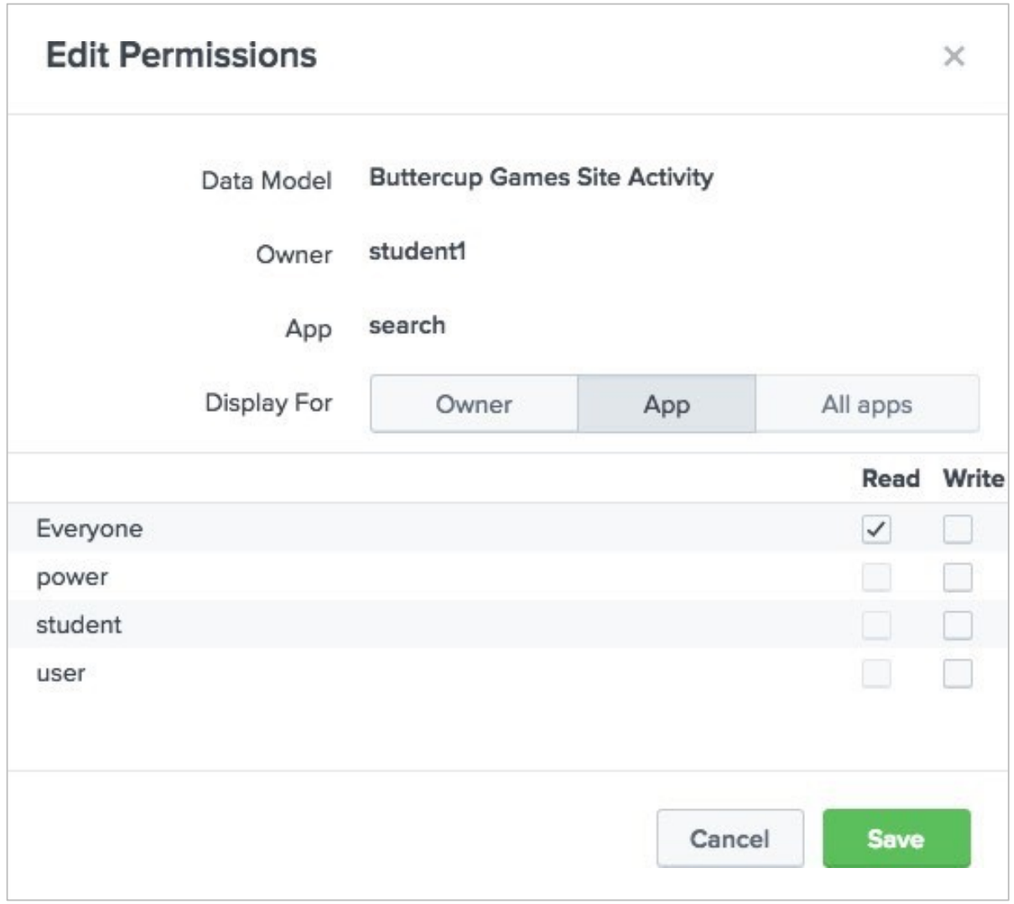


Search and Transaction Dataset Considerations

- There must be at least one event or search dataset before adding a transaction dataset
- Search and transaction datasets cannot benefit from persistent data model acceleration
- As you learn to create data models, consider the types of reports your users will run
  - Can the same report be achieved with event datasets?
  - Will users need raw events or transactional data?

Set Permissions

- When a data model is created, the owner can determine access based on the following permissions:
  - Who can see the data models
    - Owner, App, or All Apps
  - Which users can perform which actions (Read/Write)
    - Everyone
    - Power
    - User
    - Admin-defined roles, if applicable

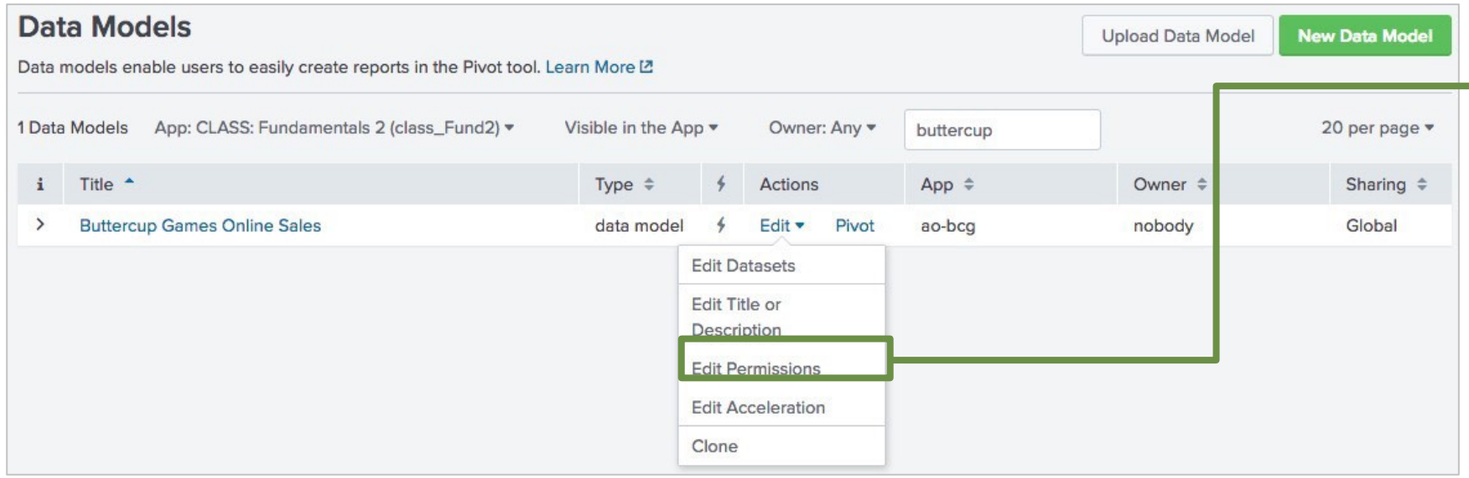


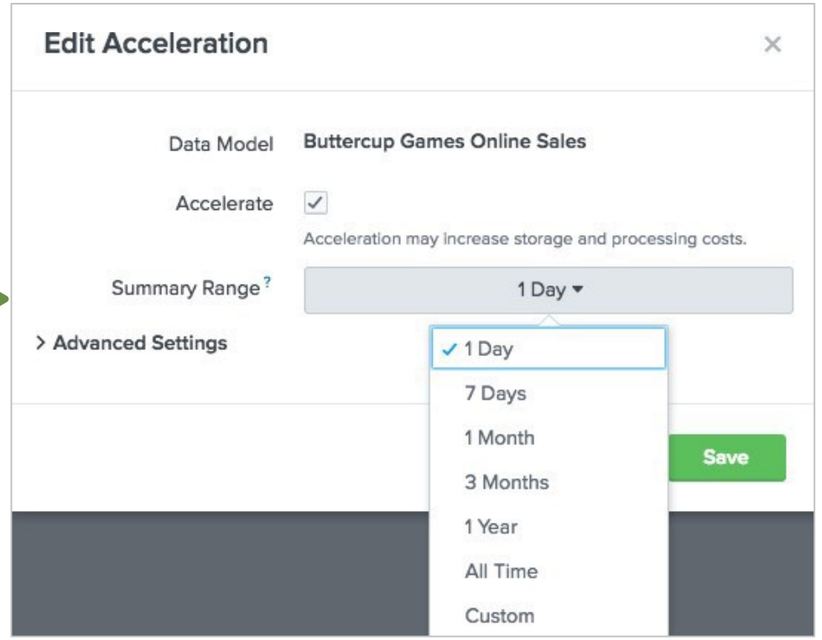
Data Model Acceleration

- Uses automatically created summaries to speed completion times for pivots
- Takes the form of inverted time-series index files (`tsidx`) that have been optimized for speed
- Discussed in more detail in *Advanced Searching and Reporting*

Accelerating a Data Model

- With persistent data model acceleration, all fields in the model become "indexed" fields
- You must have administrative permissions or the **accelerate_datamodel** capability to accelerate a data model
- Private data models cannot be accelerated
- Accelerated data models cannot be edited



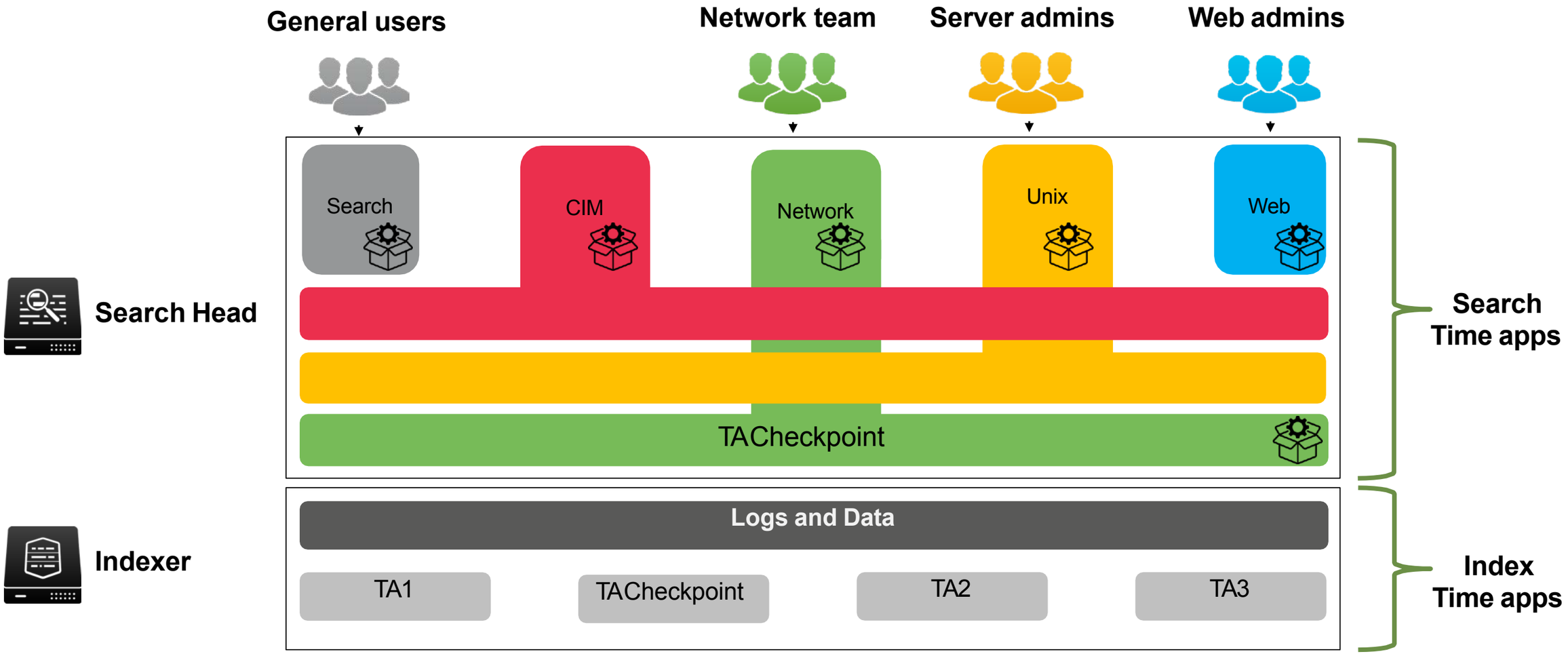


Using the Common Information Model (CIM)Add-On

What is the Common Information Model (CIM)?

- The Splunk Common Information Model provides a methodology to normalize data
- Leverage the CIM when creating field extractions, field aliases, event types, and tags to ensure:
  - Multiple apps can co-exist on a single Splunk deployment
  - Object permissions can be set to global for the use of multiple apps
  - Easier and more efficient correlation of data from different sources and source types

How the Splunk CIM Works



Normalized Field Names – Email Data

Field name	Data type	Description	Possible values
action	string	Action taken by the reporting device.	delivered, blocked, quarantined, deleted, unknown
duration	number	The amount of time for the completion of the messaging event, in seconds.	Email
src	string	The system that sent the message. May be aliased from more specific fields such as src_host, src_ip, or src_name.	

Normalized Field Names – Network Traffic

Field name	Data type	Description	Possible values
action	string	The action taken by the network device.	allowed, blocked, dropped, unknown
bytes	number	Total count of bytes handled by this device/interface (bytes_in + bytes_out).	
bytes_in	number	How many bytes this device/interface received.	
bytes_out	number	How many bytes this device/interface transmitted.	
src	string	The source of the network traffic (the client requesting the connection). May be aliased from more specific fields such as src_host, src_ip, or src_name.	

Normalized Field Names – Web Data

Field name	Data type	Description	Possible values
action	string	The action taken by the server or proxy.	
duration	number	The time taken by the proxy event, in milliseconds.	
http_method	string	The HTTP method used in the request.	GET, PUT, POST, DELETE, etc.
src	string	The source of the network traffic (the client requesting the connection).	
status	string	The HTTP response code indicating the status of the proxy request.	404, 302, 500, and so on.

Splunk CIM Add-on

- Set of 22 pre-configured data models
  - Fields and event category tags
  - Least common denominator of a domain of interest
- Leverage the CIM so that knowledge objects in multiple apps can co-exist on a single Splunk deployment
- Available on splunkbase:
  - https://splunkbase.splunk.com/app/1621/
- Use the CIM Reference Tables
  - https://docs.splunk.com/Documentation/CIM/4.9.0/User/Howtousethesereferencetables

Splunk CIM Add-On Data Models	
Alerts	Java Virtual Machines (JVM)
Application State	Malware
Authentication	Network Resolution (DNS)
Certificates	Network Sessions
Change Analysis	Network Traffic
CIM Validation (S.o.S)	Performance
Databases	Splunk Audit Logs
Email	Ticket Management
Interprocess Messaging	Updates
Intrusion Detection	Vulnerabilities
Inventory	Web

`datamodel` Command

- Search against a specified data model object
- Return a description of all or a specified data model and its objects
- Is a generating command and should be the first command in the pipeline

datamodel Learn More

Allows user to examine data models and run the search for a datamodel object.

Example:
| datamodel

datamodel Command – Example

```
| datamodel Web Web search | fields Web*
```

- Command
- Data model name
- Data model dataset name
- Command
- Find field names with Web prefix

Dataset name prepended to field names in your data

from Command

- Retrieves data from a data model or named dataset
- Must be the first command in a search
- Different than just using datamodel
  - datamodel returns **all** fields prepended with data model name
  - from datamodel returns specified fields only

```
| from datamodel:"internal_server.splunkdaccess"
```

- from can also retrieve data from saved searches, reports, or lookup files

```
| from savedsearch:mysecurityquery
```

Additional CIM Resources

- Understand and use the CIM Add-on

http://docs.splunk.com/Documentation/Splunk/latest/Knowledge/UnderstandandusetheCommonInformationModel

- Overview of the Splunk CIM

http://docs.splunk.com/Documentation/CIM/latest/User/Overview

- Use the CIM to normalize data at search time

http://docs.splunk.com/Documentation/CIM/latest/User/UsetheCIMtonormalizedataatsearchtime