Forward-Looking Statements

This presentation may contain forward-looking statements regarding future events, plans or the expected financial performance of our company, including our expectations regarding our products, technology, strategy, customers, markets, acquisitions and investments. These statements reflect management's current expectations, estimates and assumptions based on the information currently available to us. These forward-looking statements are not guarantees of future performance and involve significant risks, uncertainties and other factors that may cause our actual results, performance or achievements to be materially different from results, performance or achievements expressed or implied by the forward-looking statements contained in this presentation.

For additional information about factors that could cause actual results to differ materially from those described in the forward-looking statements made in this presentation, please refer to our periodic reports and other filings with the SEC, including the risk factors identified in our most recent quarterly reports on Form 10-Q and annual reports on Form 10-K, copies of which may be obtained by visiting the Splunk Investor Relations website at www.investors.splunk.com or the SEC's website at www.sec.gov. The forward-looking statements made in this presentation are made as of the time and date of this presentation. If reviewed after the initial presentation, even if made available by us, on our website or otherwise, it may not contain current or accurate information. We disclaim any obligation to update or revise any forward-looking statement based on new information, future events or otherwise, except as required by applicable law.

In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. We undertake no obligation either to develop the features or functionalities described, in beta or in preview (used interchangeably), or to include any such feature or functionality in a future release.

Lesser Known Search Commands Part I

PLA1159C

Kyle Smith

Integration Developer | Aplura

splunk> .conf23

Me

- Integration Developer with Aplura, LLC
- Working with Splunk for ~12 years
- Written many Public Splunk Apps (on splunkbase.splunk.com)
- Current Member of the SplunkTrust™ (8 yrs)
- Wrote the “Splunk Developer’s Guide” - Introduction to Splunk App Development
- Active on answers.splunk.com, and Slack (splk.it/slack)
- Co-leader of Baltimore Usergroup, Leader Harrisburg/Central PA
- My handle is “alacercogitatus” or just “alacer”

Splunk

- Admin
- User
- Architect
- Evangelist
- Sales Engineer
- Anybody

You

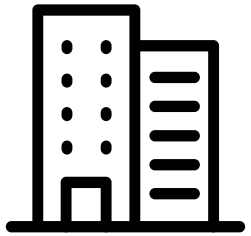
- Want to learn about new-to-you search commands
- Enjoy Piña Coladas, getting caught in the rain (well maybe not)
- Intermediate experience with SPL™
  - Do you know what “stats” does ?
  - Can you search for events?

Goals

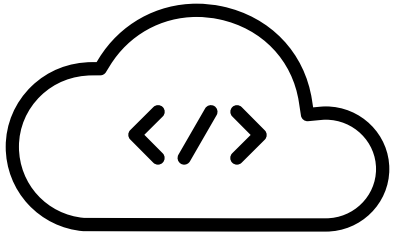
- Show/expose you to possibly new commands
- Won’t become “expert” on these commands
- Take actionable items back to your business to “try new things”

Datasets

- Internal Splunk Data
  - index IN (_internal, _audit)



- Public dataset
  - Used ‘getwatchlist’
  - Pulled in and saved via collect
  - “Film Locations in San Francisco”
    - https://data.sfgov.org/Culture-and-Recreation/Film-Locations-in-San-Francisco/yitu-d5am



```
| getwatchlist csv url=https://data.sfgov.org/resource/yitu-d5am.csv
| collect sourcetype=film_locations addtime=true
```

[1] https://splunkbase.splunk.com/app/635

Administrative (Generating) Commands

rest, makeresults, metasearch, metadata

splunk> .conf23

rest

The rest command reads a Splunk REST API endpoint and returns the resource data as a search result.[1]

- MUST be the first search command in a search block
- Is “time agnostic” - It only queries - so time is not a factor in execution
- Limits results to what the requesting user is allowed to access
- Splunk Cloud - Restricted to Search Head ONLY
- Additional Parameters are supported (see the endpoint doc)
- API Reference: https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTprolog

Why rest ?

rest can give quick insights into internal Splunk configurations, and can be used to track changes (using lookup state techniques), current configurations, and more

[1] https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Rest

rest

```
| rest /services/data/indexes splunk_server=local count=0
| table title frozenTimePeriodInSecs maxTotalDataSizeMB totalEventCount
| eval frozenTimePeriodInSecs = tostring(frozenTimePeriodInSecs, "duration")
| eval maxTotalDataSizeMB = tostring(maxTotalDataSizeMB, "commas")
| eval totalEventCount = tostring(totalEventCount, "commas")
```

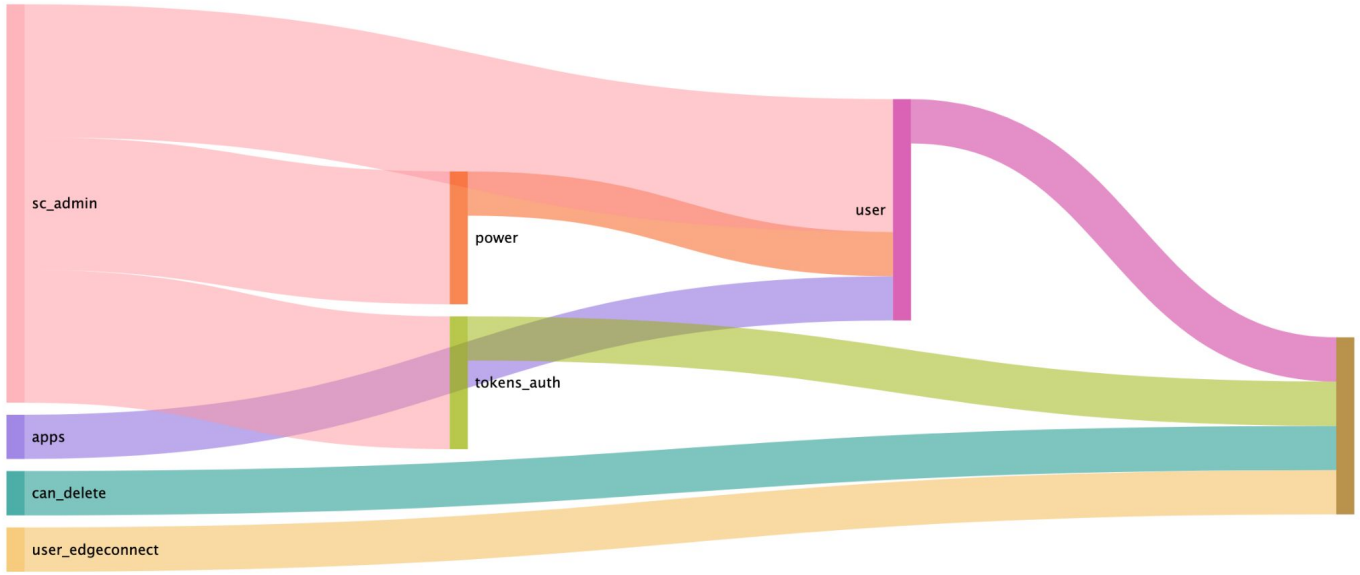
This **rest** command pulls the endpoint in question (/services/data/indexes) from the local search head, tables the data, and then formats the strings according the requirements of data type.

title	frozenTimePeriodInSecs	maxTotalDataSizeMB	totalEventCount
_audit	2184+00:00:00	500,000	13,966
_configtracker	30+00:00:00	500,000	244
_internal	30+00:00:00	500,000	71,997
_introspection	14+00:00:00	500,000	11,947
_telemetry	730+00:00:00	500,000	4

rest

```
| rest /servicesNS/-/-/authorization/roles splunk_server=local count=0
| mvexpand imported_roles
| eval count = 1
| table title imported_roles count
| eventstats count by title
```

This **rest** command pulls all authorization roles from every namespace and owner, expands the **imported_roles** (making one event per **title/imported_roles** field value), and then counts all occurrences of **title** as a column count. Graphed using **SanKey diagram**, the precedence of roles can be shown.



[1] https://splunkbase.splunk.com/app/3112

makeresults

Generates the specified number of search results. If you do not specify any of the optional arguments, this command runs on the local machine and generates one result with only the _time field.[1]

- MUST be the first search command in a search block
- Is “time agnostic” - It only creates results - so time is not a factor in execution
- Can be ‘preloaded’ with either csv or json data
- Fast, lightweight

Why makeresults ?

makeresults can quickly create fake data to evaluate a new or complicated SPL command structure, making SPL development more efficient

[1] https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Makeresults

makeresults

```
| makeresults
| eval evt=split("John;34,Sarah;23",",")
| mvexpand evt
| eval f = split(evt,";"), name=mvindex(f,0), age=mvindex(f,1)
| fields - evt f _time
```

This **makeresults** command creates an initial single event, splits apart a specially formatted string, expands the field into multiple results, and then creates additional fields from the initial string.

age	name
34	John
23	Sarah

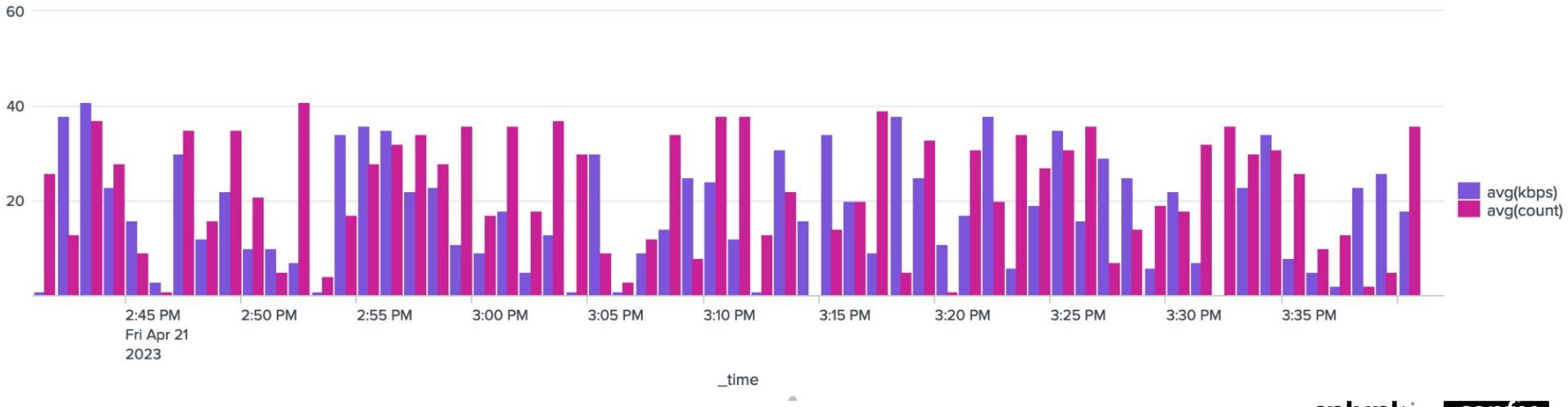
```
|makeresults format=csv data="name, age
John,35
Sarah,39"
```

```
|makeresults format=json data="[{\"name\":\"John\", \"age\":35}, {\"name\":\"Sarah\", \"age\":39}]"
```

makeresults

```
| makeresults count=60
| streamstats count as time_mod
| eval _time = now() - (60 * time_mod)
| eval kbps = ( random() % 42 ), count = ( random() % 42 )
| fields - time_mod
| timechart span=1m avg(kbps) avg(count)
```

This **makeresults** command creates 60 ‘events’, sets an increasing counter, adjusts the **_time**, generates random values for **kbps** and **count**, and then **timecharts** the averages.



metadata

The metadata command returns a list of source, sourcetypes, or hosts from a specified index or distributed search peer.[1]

- MUST be the first search command in a search block
- Useful for determining what is located in the indexes, based on metadata
- Does NOT present raw data
- Does respect the time picker, however snaps to the bucket times of the found event

Why metadata ?

metadata can give you quick event counts and times for your data.
This allows additional monitoring on incoming data, and can be used to alert if a required data source stops indexing correctly

[1] https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Metadata

metadata

```
| metadata index=_internal type=sourcetypes
| convert ctime(*Time) as *Time
| sort - totalCount
| head 10
```

This **metadata** command queries the buckets according to the time-picker, and returns the counts by **sourcetype**, sorts them, and returns the 10 largest sourcetypes. **firstTime** is the earliest “event time”, **lastTime** is the most recent “event time”, and **recentTime** is the most recent “index time”

firstTime	lastTime	recentTime	sourcetype	totalCount	type
04/19/2023 22:18:33	04/21/2023 16:30:30	04/21/2023 16:30:31	splunkd	1062660	sourcetypes
04/20/2023 08:10:28	04/21/2023 16:30:31	04/21/2023 16:30:31	splunk_btool	342612	sourcetypes
04/20/2023 08:10:53	04/21/2023 16:30:30	04/21/2023 16:30:33	splunkd_access	89315	sourcetypes
04/20/2023 08:10:54	04/21/2023 16:30:30	04/21/2023 16:30:30	splunk_assist_internal_log	16301	sourcetypes
04/20/2023 08:10:55	04/21/2023 16:30:31	04/21/2023 16:30:32	splunkd_ui_access	6070	sourcetypes
04/20/2023 08:10:36	04/21/2023 16:30:15	04/21/2023 16:30:15	secure_gateway_app_internal_log	3884	sourcetypes
04/20/2023 07:00:02	04/21/2023 16:08:27	04/21/2023 16:08:27	python_upgrade_readiness_app	1901	sourcetypes
04/20/2023 08:10:54	04/21/2023 16:02:15	04/21/2023 16:02:16	splunk_python	561	sourcetypes
04/20/2023 08:10:54	04/21/2023 16:02:15	04/21/2023 16:02:17	splunk_web_service	542	sourcetypes
04/20/2023 08:10:50	04/21/2023 15:57:14	04/21/2023 15:57:17	mongod	445	sourcetypes

metadata

```
| metadata index=_internal type=sourcetypes
| eval _time=now(), value = now() - recentTime, status = case(value>=3600,"red",value>1800,"yellow",true(),"green")
| eval threshold_warning = 1800, threshold_critical=3600
| where value>=threshold_warning
| rename sourcetype as name
| table _time, name, value, status, threshold_warning, threshold_critical
| sort - value
```

This **metadata** command queries for sourcetypes, sets a status using the difference in **now** and **recentTime**, with thresholds, and then tables the information. Uses the **Performance Analysis** visualization.

Performance Analysis Format Trellis

[1] https://splunkbase.splunk.com/app/4258

metasearch

Retrieves event metadata from indexes based on terms in the <logical-expression>. Metadata fields include source, sourcetype, host, _time, index, and splunk_server.[1]

- MUST be the first search command in a search block
- Useful for determining what is located in the indexes, based on raw data
- Does NOT present raw data
- Can only search on raw data, no extracted fields or segmenters (major or minor)
- Can be tabled based on the metadata present
- Respects the time picker and default searched indexes

Why metasearch and not tstats ?

metasearch can use raw keywords (non-segmented) to find specific parts of the data.
tstats uses aggregate functions only (no keyword search)

[1] https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Metasearch

metasearch

```
| metasearch index=_internal TERM(127.0.0.1)
| stats count by sourcetype
| sort - count
```

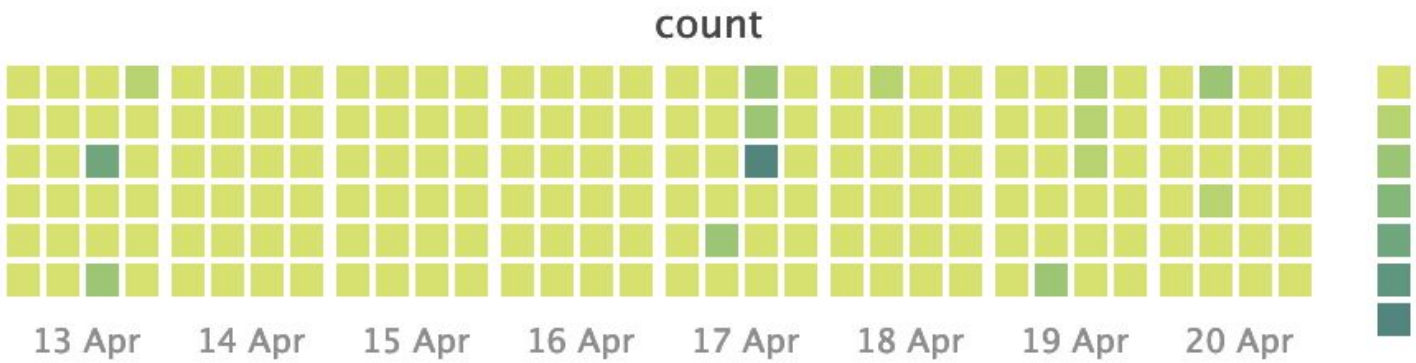
This **metasearch** command searches the _internal index for a full match (TERM) of “127.0.0.1”, counts by sourcetype and then sorts.

sourcetype	count
splunkd_access	66278
splunk_web_access	114
splunkd	8
splunk_web_service	6
splunk_python	5
splunkd_ui_access	2

metasearch

```
| metasearch index=_internal TERM(192.168.1.44) earliest=-8d@d latest=@d
| timechart span=1h count
```

This **metasearch** command searches for any exact match of “192.168.1.44” for the 7 days prior to today, and counts the hits in 1 hour buckets. Uses the **Calendar Heat Map** visualization.



[1] https://splunkbase.splunk.com/app/3162

The Difference

This search shows the differences on a non-internal index between event counts for **dbinspect**, **metadata**, **tstats**, and **metasearch**. Time picker was “last 4 hours”

```
| union
    [ metadata index=_internal type=sourcetypes
    | convert ctime(*Time) as *Time
    | eval index="_internal" , time="metadata"
    | stats sum(totalCount) as totalCount by index time]
    [ tstats count as totalCount where index=_internal by index
    | eval time = "tstats_time_picker" ]
    [ metasearch index=_internal
    | stats count as totalCount by index
    | eval time="metasearch_time_picker"]
| sort - totalCount
```

index	time	totalCount
_internal	metadata	24114083
_internal	tstats_time_picker	954638
_internal	metasearch_time_picker	954638

Iterative Commands

foreach, map

Iterative Commands. foreach, map.

splunk> .conf23

foreach

Runs a templated streaming subsearch for each field in a wildcarded field list.[1]

- Performs the same command subsearch on multiple fields (or field values)
  - Normal mode: Iterates over field list
  - Multivalue Mode: Iterates over values of a multi-value field
  - JSON Mode: Iterates over values in a JSON Array
- Can help calculate complex and repetitive tasks
- Reduces the number of evals required

PROCEED WITH CAUTION

foreach can be very expensive from a SPL™ perspective.
Iterating over **"*"** is not recommended and can cause search head failure.

[1] https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Foreach

foreach

```
index=summary sourcetype=film_locations distributor="Paramount Pictures"
| foreach actor_*
    [ eval actors = mvappend(actors, <<FIELD>>) ]
| stats dc(actors) as distinct_actors by title
| sort distinct_actors
```

This **foreach** command iterates over **actor_*** and adds each value to the **actors** field, which then allows for **stats** counting and sorting.

title	distinct_actors
A Smile Like Yours	2
Another 48 Hours	2
Fat Man and Little Boy	2
Forrest Gump	2

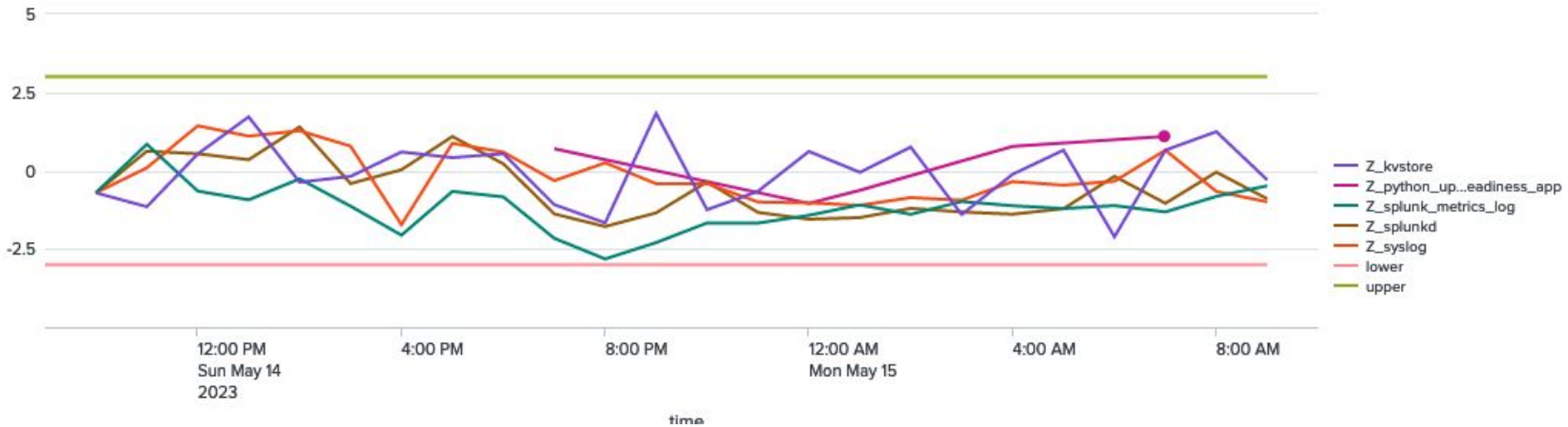
```
title="Forrest Gump", writer="Eric Roth", actor_1="Tom Hanks", actor_2="Robin Wright",
director="Robert Zemeckis", fun_facts="The original Palace was built for the 1915
Panama-Pacific Exposition, and completely destroyed in 1964. It was rebuilt in 1965.",
locations="Palace of Fine Arts (3301 Lyon Street)", production_company="Paramount
Pictures", distributor="Paramount Pictures", release_year=1994
```

foreach

```
index=_internal sourcetype=splunkd component=Metrics group=per_sourcetype_thruput
| eval ser = replace(series, ":", "-")
| timechart span=60m avg(kbps) as avg_kbps by ser useother=f
| streamstats window=720 mean(*) as MEAN* stdev(*) as STDEV*
| foreach *
    [ eval Z_<<FIELD>> = ( <<FIELD>> - MEAN<<MATCHSTR>> ) / STDEV<<MATCHSTR>> ]
| fields _time Z*
| eval upper=3, lower=-3
```

This **foreach** command analyzes sourcetype throughput, to detect for anomalies. A **timechart** with the **avg_kbps** of each sourcetype leads into a **streamstats** to get the mean and standard deviation for the rolling 720 window.

The **foreach** command iterates over any of the found fields (sourcetypes), and calculates the Z-Score.



map

The map command is a looping operator that runs a search repeatedly for each input event or result.[1]

- Uses fields from the search to create a new search and executes each new search
- Uses the same time as the picker, unless overwritten in the search
- Can be used to iterate a saved search
- Default iterations is 10, due to how expensive this gets
- Best used with ‘tabled’ data (stats, timechart) instead of raw events

PROCEED WITH CAUTION

map can be very expensive from a SPL perspective.
map is also RISKY, data modification or loss can occur

[1] https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Map

map

```
| rest /servicesNS/-/-/apps/local
| table title
| eval sourcetype="splunkd"
| map maxsearches=1000 search="rest /servicesNS/-/$title$/properties/props/$sourcetype$
        | eval app=\"$title$\", sourcetype=\"$sourcetype$\"
        |rename title as key"
| stats count by app key  | sort key app
```

This **map** command pulls app information from the API, sets a **sourcetype**, and then searches the API for **props.conf** file configurations within each app, outputting the **app** and **key** names.

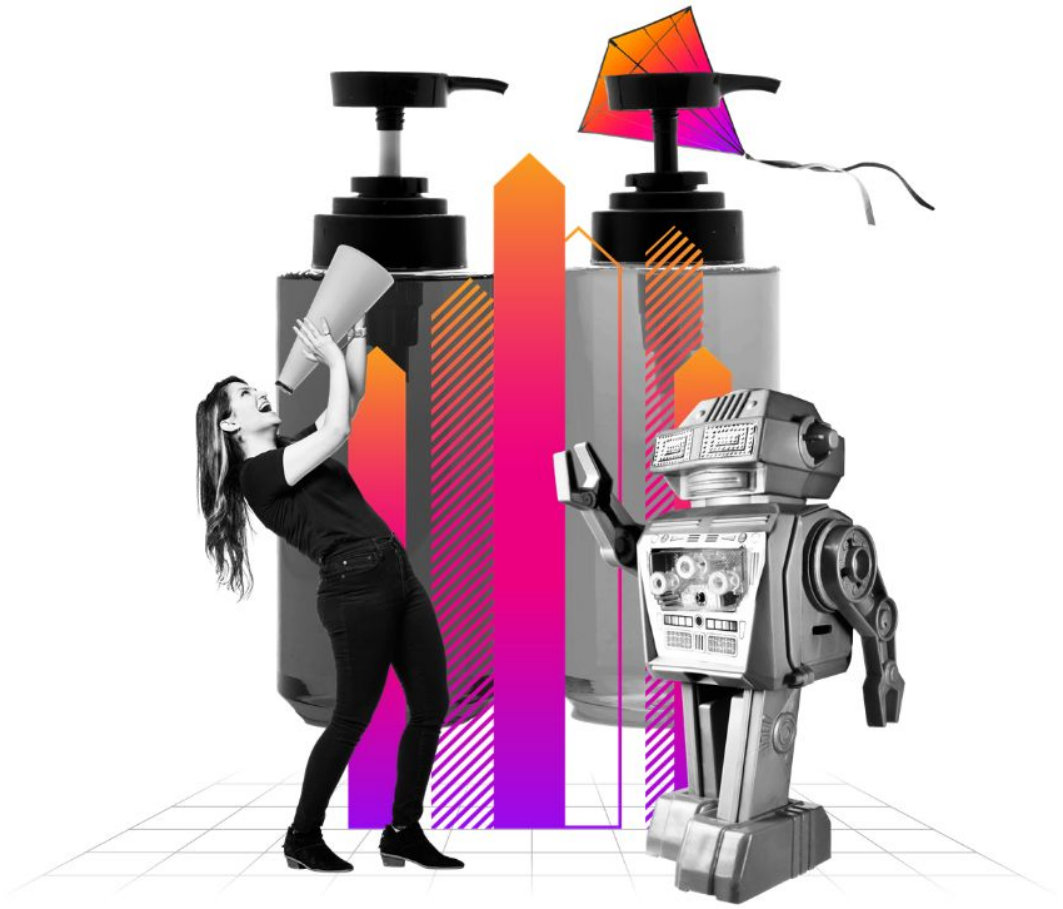
app	key	count
splunk_monitoring_console	EXTRACT-bundle_dir_reaper_max_ms	1
splunk_monitoring_console	EXTRACT-bundle_dir_reaper_mean_ms	1
splunk_monitoring_console	EXTRACT-compute_search_quota_max_ms	1
splunk_monitoring_console	EXTRACT-compute_search_quota_mean_ms	1
splunk_monitoring_console	EXTRACT-dispatch_dir_reaper_max_ms	1
splunk_monitoring_console	EXTRACT-dispatch_dir_reaper_mean_ms	1
splunk_monitoring_console	EXTRACT-enqueue_searches_count	1

```
[map]: Unexpected status for to fetch REST endpoint
uri=https://127.0.0.1:8089/servicesNS/-/alert_logevent/properties/props/splunkd?count=0
from server=https://127.0.0.1:8089 - Not Found
```

map

```
| rest /servicesNS/-/-/apps/local
| table title
| eval sourcetype="splunkd"
| map maxsearches=1000 search="rest /servicesNS/-/$title$/properties/props/$sourcetype$
                |eval app=\"$title$\", sourcetype=\"$sourcetype$\"|rename title as key"
| appendpipe
   [ map maxsearches=1000 search="rest /servicesNS/-/$app$/properties/props/$sourcetype$/$key$
                |eval app=\"$app$\", key=\"$key$\" " ]
| stats values(value) as value by app key
| eventstats count(key) as key_count by key
| sort - -key_count key app
```

app	key	value	key_count
search	EXTRACT-fields	(?i)^(?:[^]*){2}(?:[+\-]\d+)?(?P<log_level>[^]*)\s+ (?P<component>[^]+) (?:\[(?P<thread_id>\d+)\s)? (?:(?P<thread_name>[^\]]+)\]\s)?- (?P<event_message>.+)	2
Splunk_SA_CIM	EXTRACT-fields	(?i)^(?:[^]*){2}(?:[+\-]\d+)?(?P<log_level>[^]*)\s+ (?P<component>[^]+) (?:\[(?P<thread_id>\d+)\s)? (?:(?P<thread_name>[^\]]+)\]\s)?- (?P<event_message>.+) (?P<additional_info>.*)?	2
Splunk_SA_CIM	REPORT-signature_for_sendmodalert	signature_for_sendmodalert	1



Statistical Commands

delta, xyseries, untable, timewrap

delta

Computes the difference between nearby results using the value of a specific numeric field.[1]

- Very specific uses
- Cannot be used on multiple series - has no concept of ‘by’ clause
- Numerical (# date_hour) fields only
- Using p>1 with multiple **delta** commands can get weird (`| delta _time as delta_time | delta p=3 ev as delta_ev`)

Why delta?

delta can “skip” rows/events to perform the calculation, removing the need for complex **streamstats** or **eval**s

[1] https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Delta

delta

```
index=_internal sourcetype=splunkd component=Metrics
        group=per_sourcetype_thruput series=splunkd_ui_access
| reverse
| table _time ev
| delta ev as delta_ev
| delta _time as delta_time
| eval delta_eps = delta_ev / delta_time
```

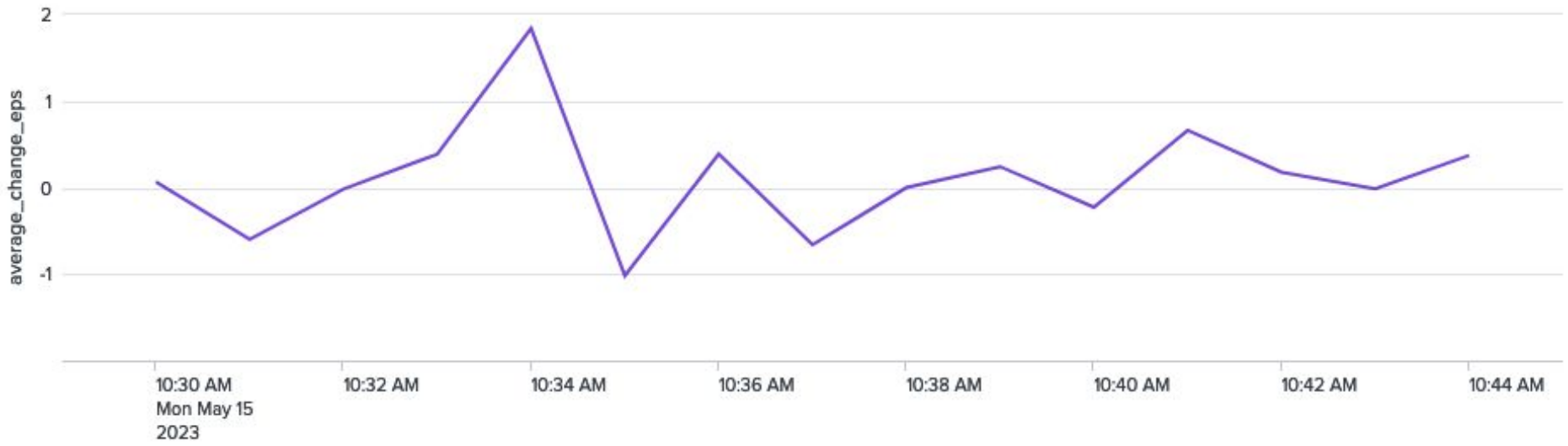
This **delta** command is calculating the change in events per second in the UI access logs. The data must first be **reverse**d to account for the direction of time to calculate delta in time moving forward.

_time	ev	delta_eps	delta_ev	delta_time
2023-05-15 10:21:35.046	14			
2023-05-15 10:22:06.024	15	0.03	1	30.978
2023-05-15 10:22:37.001	13	-0.06	-2	30.977
2023-05-15 10:23:07.986	17	0.1	4	30.985
2023-05-15 10:23:38.966	15	-0.06	-2	30.980
2023-05-15 10:24:09.936	14	-0.03	-1	30.970

delta

```
index=_internal sourcetype=splunkd component=Metrics
        group=per_sourcetype_thruput series=splunkd_ui_access
| reverse
| table _time eps
| delta eps as delta_eps
| timechart span=1m avg(delta_eps) as average_change_eps
```

This **delta** command calculates change in events per second, and displays the average over 1 minute.



xyseries

Converts results into a tabular format that is suitable for graphing.[1]

- Optional arguments to finesse the data
- Alias to **maketable**
- Results with duplicate values are removed
- Great for post-processing searches within dashboards

UNTABLE Opposite

xyseries does the opposite of **untable**, which will be covered next.

[1] https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Xyseries

xyseries

```
index=summary sourcetype=film_locations
| stats dc(locations) as locations by distributor release_year
| where locations > 10
| xyseries release_year distributor locations
| sort release_year
```

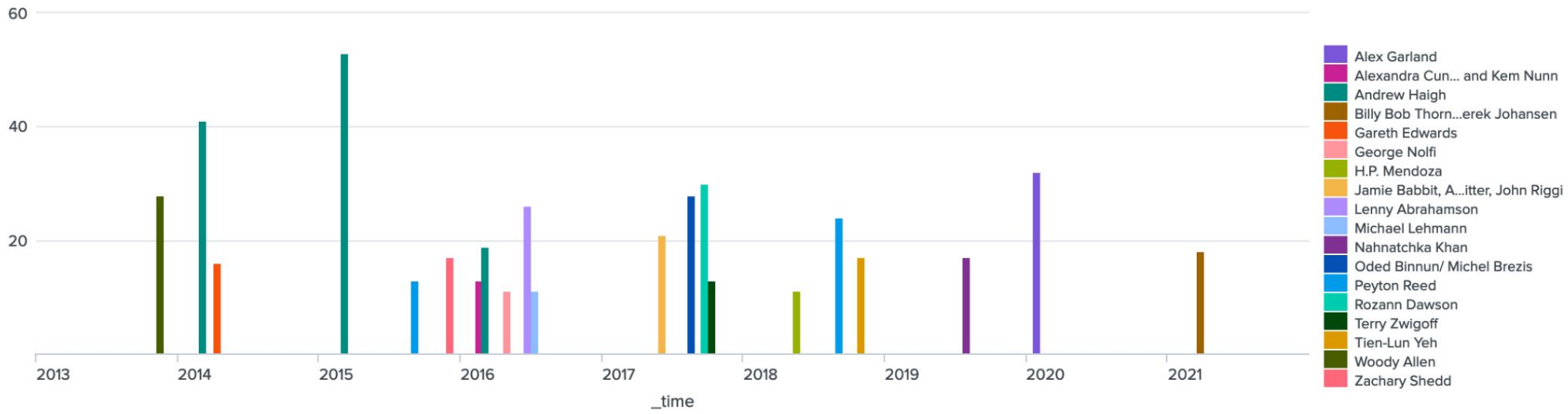
This **xyseries** command runs after a stats command that is count distinct locations by who distributes the move, and from what year the movie was from, then filter out movies without at least 10 locations.

release_year	ABS-CBN, Gravitas Ventures	Amazon	Amazon Studios	American Broadcasting Company (ABC)	Dimension Films	FX Network	HBO	HULU	Home Box Office (HBO)	Hulu	Kylin Pictures	Metro-Goldwyn-Mayer (MGM)
1968												
1971												
1973												
1974				12								
1978												
1992												
1994												11
2000					12							
2013												
2014									41			
2015							53					
2016							19	71			11	

xyseries

```
index=summary sourcetype=film_locations
| stats dc(locations) as locations by director release_year
| where locations > 10 AND release_year >= 2003
| eval _time = strptime(release_year."-01-01", "%Y-%m-%d")
| xyseries _time director locations
```

This **xyseries** command uses a filtered set of distinct counts of locations by director and release year over the last 20 years.



untable

Converts results from a tabular format to a format similar to stats output.[1]

- Results with duplicate values are removed
- Great for post-processing searches within dashboards
- Simple, yet effective
- Only 3 arguments

XYSERIES Opposite

untable does the opposite of **xyseries**

[1] https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Untable

untable

```
index=summary sourcetype=film_locations
| stats count by director distributor release_year
| untable director FieldName FieldValue
```

This **untable** command counts the events by director, distributor, and release_year, then untables the data into 3 fields. Each of the fields becomes a value of the field named ‘FieldName’

director	FieldName	FieldValue
Alan Jacobs	count	1
Alan Jacobs	distributor	First Look International
Alan Jacobs	release_year	2000
Alex Garland	count	32
Alex Garland	distributor	FX Network
Alex Garland	release_year	2020
Alexandra Cunningham and Kem Nunn	count	13
Alexandra Cunningham and Kem Nunn	distributor	HULU

untable

```
index=_internal log_level=ERROR
| timechart span=15m count by component
| untable _time name value
| eval status = "ok", threshold_warning = 1, threshold_critical = 3
| table _time name value status threshold_warning threshold_critical
```

This **untable** command takes the output from a **timechart**, and creates a new table that has the **component** in the **name** field, and the **count** in the **value** field.

Uses the **Performance Analysis** visualization on Splunkbase.

[1] https://splunkbase.splunk.com/app/4258

timewrap

Displays, or wraps, the output of the timechart command so that every period of time is a different series.[1]

- Much easier than using complicated evals
- Won't calculate until entire previous SPL complete
- Modifies the dates of events for display
- Useful to show change over periods (YoY, MoM)

Limit the number of fields

timewrap works best with the **timechart** command, with limited number of fields. For each **timechart** field, there are N times number of fields depending on the **timewrap** span

[1] https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Timewrap

timewrap

```
index=_internal sourcetype=splunkd component=Metrics
      group=per_index_thruput series="_audit" earliest=-4w@w latest=@w+1d
| eval series = "audit_index"
| timechart span=1d sum(kb) as size by series
| timewrap 1week
```

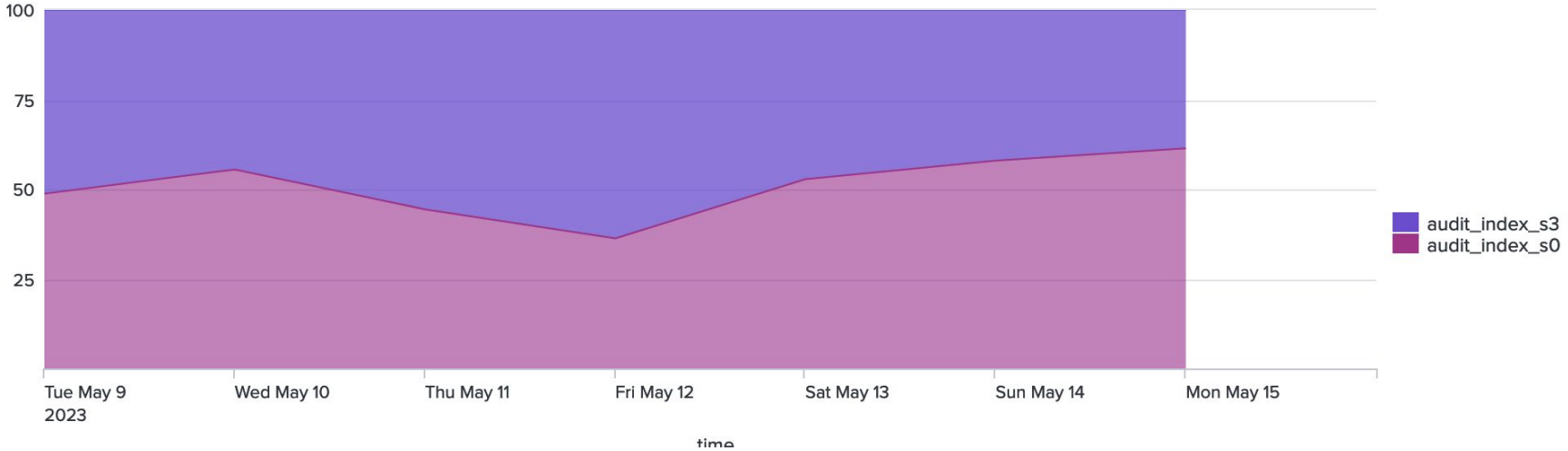
This **timewrap** command takes the output of a **timechart** (by 1d), and wraps the results in 1 week “buckets”, updates the field names, and adjusts the **_time**

_time	audit_index_4weeks_before	audit_index_3weeks_before	audit_index_2weeks_before	audit_index_1week_before
2023-05-08		2229.803	2747.773	2051.486
2023-05-09		2073.050	2447.900	1972.277
2023-05-10		1580.336	1775.051	2541.126
2023-05-11		2474.504	1774.008	1975.978
2023-05-12		3472.751	2632.198	2094.425
2023-05-13		1763.997	3127.116	1972.934
2023-05-14	1720.985	2041.958	2003.359	1986.659

timewrap

```
| union
    [ search index=_internal sourcetype=splunkd component=Metrics
        group=per_index_thruput series="_audit" earliest=-4w@w+2d latest=-3w@w+2d]
    [ search index=_internal sourcetype=splunkd component=Metrics
        group=per_index_thruput series="_audit" earliest=-1w@w+2d latest=@w+2d]
| eval series = "audit_index"
| timechart span=1d sum(kb) as size by series
| timewrap 1week series=short
```

This **timewrap** command uses two different internal searches, **timecharts** the results, and sets the fields to short names. Chart is column, 100% stacked to show variation between weeks.



GITHUB

https://github.com/alacercogitatus/pla1159C-supplemental-app

Part II

- See me in 40 years!
- SEE: arules, associate, correlate, contingency
- SEE: tstats, streamstats, eventstats, and maybe more!

Go forth and SPL™ yourself!

- Administrative: rest, makeresults, metadata, metasearch
- Iterative: foreach, map
- Statistical: delta, xyseries, untable, timewrap

@.conf23 - Where to next?

Listen to me as I describe where to go next!

- PLA1547B - Lighter, Faster and Calmer Ways to Learn Splunk® Enterprise With | makeresults, | gentimes and Some Random()% Too!
- PLA1577B - Dashboarding Wowzas! Top Tips for Making Your Dashboards Awesome!
- PLA1765C - Git Good With Splunk: Commit to Config Versioning and Deployment Automation for Your Splunk Infrastructure
- PLA1881C - Maximizing Splunk SPL™: Foreach and the Power of Iterative, Templatized Evals

linktr.ee/alacercogitatus
All the Docs!

Resources

- https://docs.splunk.com
- https://answers.splunk.com
- https://lantern.splunk.com
- Slack!
  - https://splk.it/slack
- Join a User Group!
  - https://usergroups.splunk.com
- https://www.splunk.community
- The Splunk Trust - We are here to help! (find us by our fez!)

Thank You

splunk> .conf23