

Lesser Known Search Commands

Kyle Smith

Integration Developer / Aplura, LLC

.conf2016

splunk>

Disclaimer

During the course of this presentation, we may make forward looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC. The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make. 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. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

Me



- Integration Developer with Aplura, LLC
- Working with Splunk for ~6 years
- Written many Public Splunk Apps (on splunkbase.splunk.com)
- Current Member of the SplunkTrust
- Wrote the “Splunk Developer’s Guide” - introduction to Splunk App Development
- Active on both #splunk on IRC and answers.splunk.com
 - My Handle is “alacer cogitatus” or just “alacer”

Agenda

- **Administrative**
 - rest, makeresults, gentimes, metasearch, metadata
- **Iterative**
 - map, foreach
- **Statistics**
 - Contingency, tstats, xyseries, streamstats, eventstats, autoregress
- **Unsupported Hacks**
 - timechart / dynamic evals

NOTE: These commands have been verified for 6.4. Any other version may or may not work correctly

Administrative Commands

- rest, makeresults, gentimes, metasearch, metadata
- Commands that do not pull full (raw) events or are used to interact with Splunk
- Provide information about data or the Splunk server(s)

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

```
| rest /services/data/indexes  
splunk_server=local count=0 | dedup title |  
fields title
```

title
_audit
_internal
_introspection
_thefishbucket
apps
firealerts
history
httpd
main
minecraft
minecraft_madscience

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

rest

```
|rest /services/authentication/current-context splunk_server=local | stats values(roles) as roles values(defaultApp) as defaultApp by username
```

This endpoint pulls the currently logged in user's authorization context. With this information, you could restrict specific dashboards or searches by role, even more granular than the pre-built permissions.

defaultApp	username	roles
launcher	ksmith	admin windows-admin winfra-admin
launcher	splunk-system-user	admin splunk-system-role

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

- New in 6.3
- Easy way to “spoof” data to experiment with evals, and other SPL commands
- Fast, lightweight
- Use it to restrict a search using it in a subsearch

```
index=_internal _indextime > [makeresults | eval it=now()-60 | return $it]
```

<code>_time</code> ▾	it ▾
2016-07-26 11:25:52	1469546692

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

gentimes

Generates timestamp results starting with the exact time specified as start time. Each result describes an adjacent, non-overlapping time range as indicated by the increment value. This terminates when enough results are generated to pass the endtime value.¹

- Useful for generating time buckets not present due to lack of events within those time buckets
- Must be the first command of a search (useful with map, or)
- “Supporting Search” - no real use case for basic searching

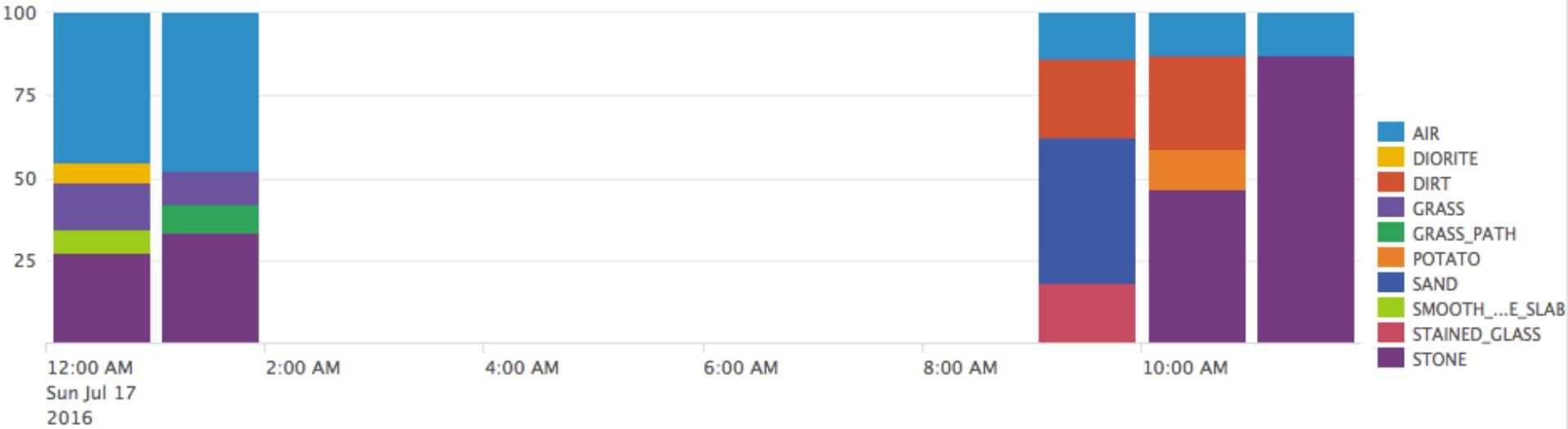
```
| gentimes start=10/1/15 end=10/5/15
```

endhuman ◆	endtime ◆	starthuman ◆	starttime ◆
Thu Oct 1 23:59:59 2015	1443758399	Thu Oct 1 00:00:00 2015	1443672000
Fri Oct 2 23:59:59 2015	1443844799	Fri Oct 2 00:00:00 2015	1443758400
Sat Oct 3 23:59:59 2015	1443931199	Sat Oct 3 00:00:00 2015	1443844800
Sun Oct 4 23:59:59 2015	1444017599	Sun Oct 4 00:00:00 2015	1443931200

[1] <https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Gentimes>

gentimes

```
|gentimes start=-9 end=-7 increment=1h | map maxsearches=48 search="search earliest=$starttime$ latest=$endtime$ eventtype=splunkcraft2016 | bucket _time span=1h | top useother=f limit=5 block_type by _time | fields - percent" | timechart span=1h sum(count) by block_type useother=f
```



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

- Useful for determining what is located in the indexes, based on raw data
- Does NOT present raw data
- Can be tabled based on the metadata present
- Respects the time picker and default searched indexes

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

metasearch

```
| metasearch eventtype=splunkcraft2016 earliest=-24h@h
```

The screenshot shows a Splunk search interface. On the left, there's a sidebar with 'Selected Fields' containing 'host 1', 'source 1', and 'sourcetype 1'. Below that is a section for 'Interesting Fields' with 'index 1', 'splunk_server 1', and 'splunk_server_group 5'. The main area is titled 'sourcetype' and displays the following information:

- 1 Value, 100% of events
- Selected: Yes
- Reports: Top values, Top values by time, Events with this field
- Values table:

Values	Count	%
minecraft_log	37,384	100%

metadata

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

- 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

```
| metadata type=sourcetypes | convert ctime(firstTime) timeformat="%Y-%m-%d %H:%M:%S" | convert ctime(lastTime) timeformat="%Y-%m-%d %H:%M:%S" | convert ctime(recentTime) timeformat="%Y-%m-%d %H:%M:%S"
```

firstTime	lastTime	recentTime	sourcetype	totalCount	type
2016-07-26 10:59:54	2016-07-26 13:07:02	2016-07-26 13:07:18	minecraftConsole_log	232	sourcetypes
2016-07-26 11:03:08	2016-07-26 12:18:18	2016-07-26 12:18:18	minecraft_log	84	sourcetypes

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

Iterative

- map, foreach
- Iterative commands do looping with events or with fields in the search pipeline.

map

The map command is a looping operator that runs a search repeatedly for each input event or result. You can run the map command on a saved search, a current search, or a subsearch.¹

- Uses “tokens” (\$field\$) to pass values into the search from the previous results
- Best with either: Very small input set And/Or very specific search.
- Can take a long amount of time.
- Map is a type of subsearch
- Is “time agnostic” - time is not necessarily linear, and can be based off of the passed events, if they include time

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

map

```
| rest /services/data/indexes splunk_server=local count=0 | dedup title |  
fields title | map [|metadata type=sourcetypes index="$title$" | eval  
type="$title$"] maxsearches=1000 | stats values(totalCount) by sourcetype  
type | rename type as index
```

It takes each of the results from the rest search and searches the metadata in each index.
The results are returned as a table, such as:

sourcetype	index	values(totalCount)
ActiveDirectory	msad	2509
GigamonForSplunk:error	main	6
Linux:SELinuxConfig	os	2393
MSAD:NT6:Health	msad	2250
MSAD:NT6:Replication	msad	13512
MSAD:NT6:SiteInfo	msad	380
Perfmon:CPU	perfmon	7526334

map

```
|rest /services/data/indexes count=0 | dedup title | fields title | map [ |metadata type=sourcetypes index="$title$" | eval type="$title$"] maxsearches=1000 | stats values(totalCount) by sourcetype type | rename type as index | stats count(index) as ci by sourcetype | where ci > 1
```

This can quickly show you where there may be a sourcetype misconfigured. Why would it be in two different indexes (unless permissions play a role)? The same can be done for sources and hosts.

sourcetype	ci
apache_error	2
postfix_syslog	2
syslog	2

foreach

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

- Rapidly perform evaluations and other commands on a series of fields
- Can help calculate Z scores (statistical inference comparison)
- Reduces the number of evals required

```
Equivalent to ... | eval foo="foo" | eval bar="bar" | eval baz="baz"  
... | foreach foo bar baz [eval <<FIELD>> = "<<FIELD>>"]
```

Can also use wildcards

```
| foreach foo* [ eval <<MATCHSEG1>> = "<<FIELD>>" ]  
foobar = This, foobaz = That ➔ bar = This, baz = That
```

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

foreach

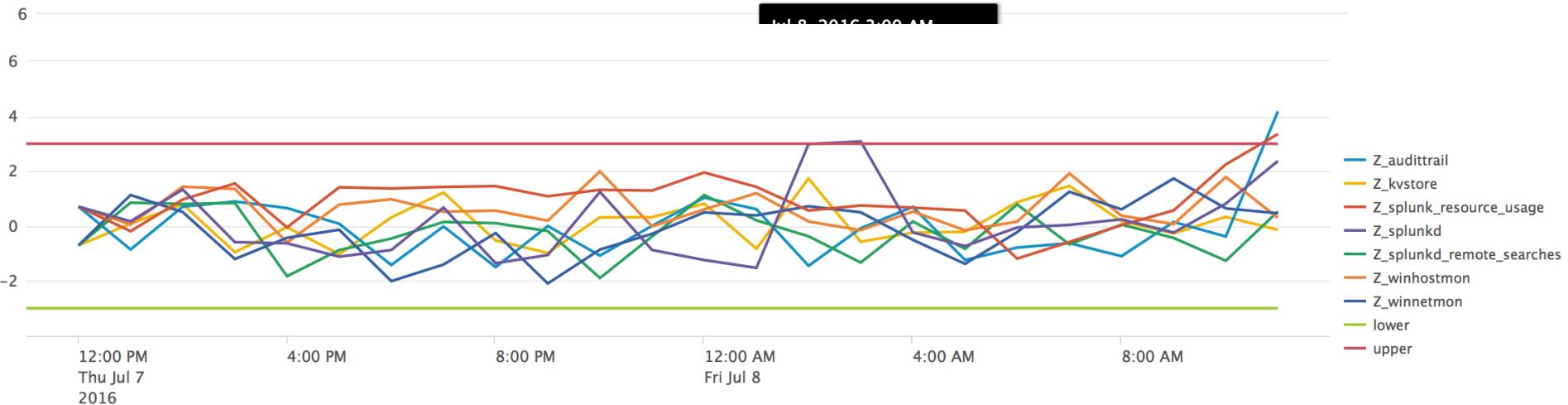
```
index=_internal sourcetype=splunkd component=Metrics group=per_sourcetype_thruput | timechart  
span=60m avg(kbps) as avg_kbps by series useother=f | streamstats window=720 mean(*) as MEAN*  
stdev(*) as STDEV* | foreach * [ eval Z_<<FIELD>> = ((<<FIELD>>-MEAN<<MATCHSTR>>) /  
STDEV<<MATCHSTR>>)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

_time	Z_audittrail	Z_kvstore	Z_splunk_resource_usage	Z_splunkd	Z_splunkd_remote_searches	Z_winhostmon	Z_winnetmon
2016-07-07 11:00							
2016-07-07 12:00	0.7071	-0.7073	0.7067	0.7070	-0.7070	0.70710	-0.70711
2016-07-07 13:00	-0.8564	0.0849	-0.189	0.171	0.84928	0.05686	1.12954
2016-07-07 14:00	0.7051	0.7910	0.9656	1.330	0.8034	1.43314	0.509439
2016-07-07 15:00	0.8939	-0.9532	1.550	-0.589	0.8402	1.35145	-1.20408
2016-07-07 16:00	0.6538	-0.046	-0.0504	-0.630	-1.8309	-0.60302	-0.430569
2016-07-07 17:00	0.0813	-1.030	1.411	-1.118	-0.87633	0.779357	-0.13692
2016-07-07 18:00	-1.421	0.314	1.369	-0.8766	-0.4585	0.973972	-2.01103
2016-07-07 19:00	-0.0174	1.220	1.425	0.6707	0.1472	0.52094	-1.40475
2016-07-07 20:00	-1.499	-0.526	1.453	-1.356	0.1022	0.56083	-0.258726

foreach



```
index=_internal sourcetype=splunkd component=Metrics group=per_sourcetype_thruput | timechart  
span=60m avg(kbps) as avg_kbps by series 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
```



Statistics

- `contingency`, `tstats`, `xseries`, `streamstats`, `eventstats`, `autoregress`
- These are commands that build tables, evaluate sums, counts or other statistical values

contingency

In statistics, [contingency tables](#) are used to record and analyze the relationship between two or more (usually categorical) variables.¹

A contingency table is a table showing the distribution (count) of one variable in rows and another in columns, and is used to study the association between the two variables.

Contingency is best used where there is a single value of a variable per event.

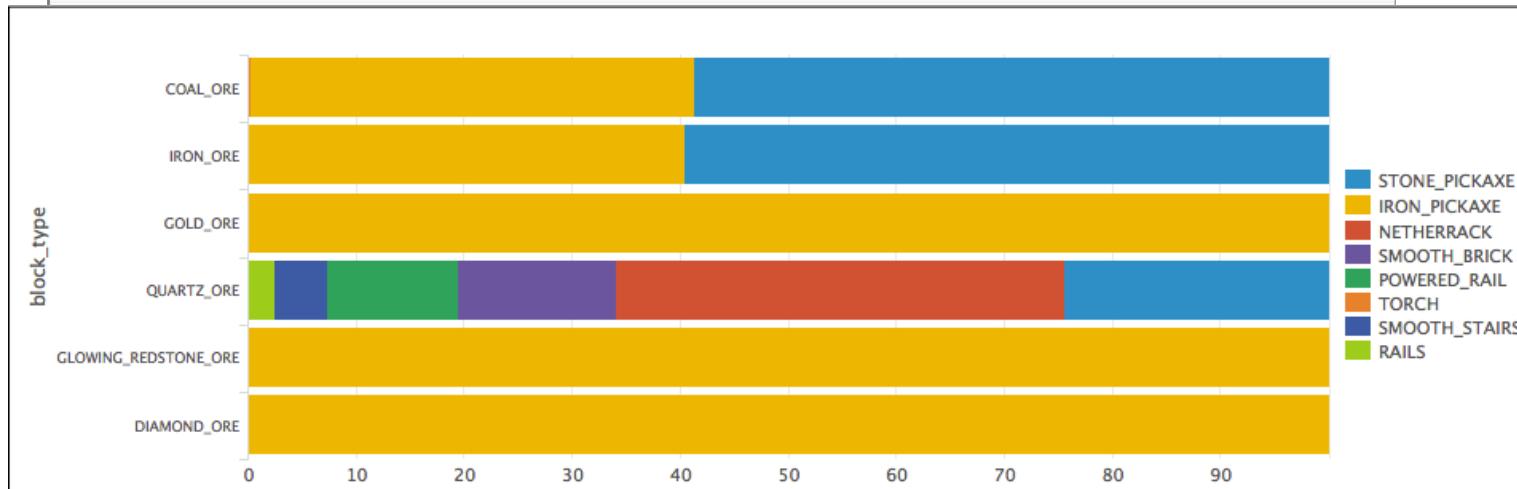
- Web Analytics - Browsers with Versions
- Demographics - Ages with Locations or Genders
- Security - Usernames with Proxy Categories
- Great to compare categorical fields

[1] <https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Contingency>

contingency

eventtype=splunkcraft2016 base_type=*ORE* | contingency block_type tool_used usetotal=f

block_type	STONE_PICKAXE	IRON_PICKAXE	NETHERRACK	SMOOTH_BRICK	POWERED_RAIL	TORCH	SMOOTH_STAIRS
COAL_ORE	501	351	0	0	0	2	0
IRON_ORE	398	270	0	0	0	0	0
GOLD_ORE	0	77	0	0	0	0	0
QUARTZ_ORE	10	0	17	6	5	0	2
GLOWING_REDSTONE_ORE	0	19	0	0	0	0	0
DIAMOND_ORE	0	13	0	0	0	0	0



tstats

Use the tstats command to perform statistical queries on indexed fields in tsidx files. The indexed fields can be from normal index data, tscollect data, or accelerated data models.¹

- Can only be used on indexed fields. EXTRACTED FIELDS WILL NOT WORK
- Quick way to access metadata or accelerated data (from data models or saved searches)
- Respects the time picker and default searched indexes

```
|tstats count by sourcetype index | stats dc(index) as ci by sourcetype
```

sourcetype	ci
minecraftConsole_log	1
minecraft_log	2
stream:dns	1
stream:http	1
stream:tcp	1
stream:tns	1

[1] <https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Tstats>

xyseries

Converts results into a format suitable for graphing.¹

Xyseries can help you build a chart with multiple data series.

- Email Flow [xyseries email_domain email_direction count]
- One to Many relationships [example Weather Icons]
- Any data that has values INDEPENDENT of the field name
- host=myhost domain=splunk.com metric=kbps metric_value=100
- xyseries domain metric metric_value
- Works great for Categorical Field comparison

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

xyseries

```
`weather_data` | xyseries icon weather weather
```

icon	Clear	Fog	Haze	Heavy Rain	Heavy Thunderstorms and Rain	Light Rain	Light Thunderstorms and Rain	Mist	Mostly Cloudy	Overcast	Partly Cloudy	Rain	Scattered Clouds	Thunderstorms and Rain
clear	Clear													
cloudy										Overcast				
fog	Fog													
hazy			Haze						Mist					
mostlycloudy									Mostly Cloudy					
partlycloudy										Partly Cloudy		Scattered Clouds		
rain			Heavy Rain		Light Rain						Rain			
tstorms				Heavy Thunderstorms and Rain		Light Thunderstorms and Rain							Thunderstorms and Rain	

eventstats

Adds summary statistics to all search results.¹

```
eventtype=splunkcraft2016 player=alacer cogitatus | eventstats  
dc(block_type) as dc_block_type by player | table player dc_block_type
```

player ◁	dc_block_type ◁
alacer cogitatus	13

[1] <https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Eventstats>

streamstats

Adds cumulative summary statistics to all search results in a streaming manner. The streamstats command calculates statistics for each event at the time the event is seen.¹

```
eventtype=splunkcraft2016  
player=alacer cogitatus |  
streamstats  
dc(block_type) as  
dc_block_type by player |  
table player  
dc_block_type
```

alacer cogitatus	13
alacer cogitatus	13
alacer cogitatus	13
alacer cogitatus	12
alacer cogitatus	11

[1] <https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Streamstats>

autoregress

Prepares your events for calculating the autoregression, or the *moving average*, by copying one or more of the previous values for *field* into each event.¹

A Moving Average is a succession of averages calculated from successive events (typically of constant size and overlapping) of a series of values.

- Allows advanced statistical calculations based on previous values
- Moving Averages of numerical fields
- Network bandwidth trending - kbps, latency, duration of connections
- Web Analytics Trending - number of visits, duration of visits, average download size
- Malicious Traffic Trending - excessive connection failures

[1] <https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Autoregress>

autoregress

```
eventtype=splunkcraft2016 | autoregress action as preaction | eval  
p_action = if(action==preaction,1,0) | stats avg(p_action) by player
```

player	avg(p_action)
JD_Warrior	0.838235
JoxerTheMighty1	0.250000
LordProducer008	0.545571
Lovelace47	0.724025
Meson3902	0.664076
acharlieh	0.648207
alacer cogitatus	0.714668
brianmiddleton	0.676733
brinman2002	0.795402
viciousbite	0.737643

Unsupported

- Eval function with a stats/timechart command
- Indirect Referencing
- These are not actual commands, but are more along the lines of a hack

Inline Timeline Eval

- You can use an eval statement in a timechart command

```
index=_internal sourcetype=splunkd source=*metrics.log group=per_sourcetype_thruput | eval  
ev2 = kb / ev | timechart span=1h eval(avg(kb) / avg(ev)) as "AVG KB per Event - 2"  
avg(ev2) as "AVG KB per Event - 1"
```

AVG KB per Event - 2	AVG KB per Event - 1
18.10579	18.105794
18.09486	18.094857
18.04806	18.048062
17.91012	17.910115
17.77742	17.777419

- There is a difference in significant digits
- Must rename the field

Dynamic Eval (aka Indirect Reference)

- Not a search command
- NOTE: It's a hack, so it might not work in the future.
- Works great for perfmon sourcetypes, but can be applied to any search

```
sourcetype=perfmon:dns earliest=-1h@h | eval cnt_{counter} = Value | stats avg(cnt_*) as *
```

The Raw Event

```
07/29/2016 06:07:01.973 -0800
collection=DNS
object=DNS
counter="TCP Message Memory"
instance=0
Value=39176
```

The New Event

```
07/29/2016 06:07:01.973 -0800
collection=DNS
object=DNS
counter="TCP Message Memory"
instance=0
Value=39176
cnt_TCP_Message_Memory = 39176
```

Dynamic Eval - Subsearch

- Not a search command
- NOTE: It's a Splunk hack, so it might not work in the future

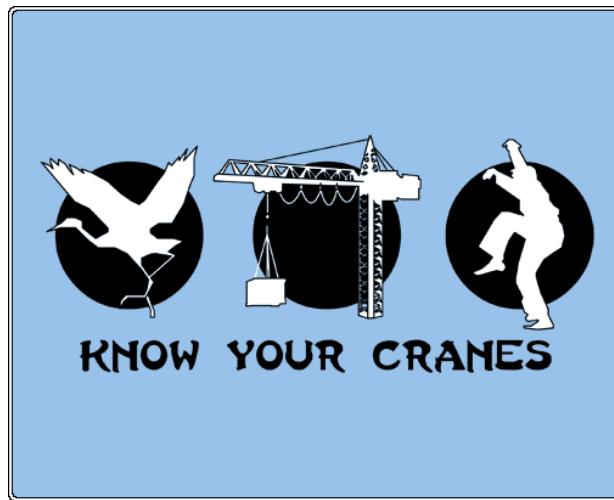
```
index=_internal sourcetype=splunkd source=*metrics.log group=per_sourcetype_thruput | eval  
sub_host = replace([ |metadata type=hosts index=_internal | head 1 | rename host as query |  
fields query | eval query=""".query.""""], """", "") | eval subsearch =  
if(host==sub_host,"setting_1","setting_2")
```

host	sub_host	subsearch
1f6cc49cc777	1f6cc49cc777	setting_1



CLI Commands

- Commands that are run from the command line to help extract data, pull configurations, etc.



CLI Commands

- `$SPLUNK_HOME/bin/splunk reload index`
 - reloads index configuration, making immediately effective all "add/edit/enable/disable index" commands since last reload or Splunk restart
- Why?
 - Adding a new app
 - Changing a frozen time period
 - New Location for data

CLI Commands

- `$SPLUNK_HOME/bin/splunk cmd pcregextest`
 - Useful for testing regular expressions for extractions

```
splunk cmd pcregextest mregex="[[ip:src_]] [[ip:dst_]]" ip="(?(?<ip>\d+[dotnum]\{3\})" dotnum="\.\.\d+" test_str="1.1.1.1 2.2.2.2"
```

Original Pattern: '[[ip:src_]] [[ip:dst_]]'

Expanded Pattern: '(?(?<src_ip>\d+(?:\.\d+)\{3\}) (?(?<dst_ip>\d+(?:\.\d+)\{3\}))'

Regex compiled successfully. Capture group count = 2. Named capturing groups = 2.

SUCCESS - match against: '1.1.1.1 2.2.2.2'

Capturing group data

Group	Name	Value
-------	------	-------

1	src_ip	1.1.1.1
2	dst_ip	2.2.2.2

CLI Commands

- `$SPLUNK_HOME/bin/splunk cmd btool`
 - Btool allows you to inspect configurations and what is actually being applied to your sourcetypes
- `splunk cmd btool --debug props list wunderground | grep -v "system/default"`
`/opt/splunk/etc/apps/TA-wunderground/default/props.conf [wunderground]`
`/opt/splunk/etc/apps/TA-wunderground/default/props.conf KV_MODE = json`
`/opt/splunk/etc/apps/TA-wunderground/default/props.conf MAX_EVENTS = 100000`
`/opt/splunk/etc/apps/TA-wunderground/default/props.conf MAX_TIMESTAMP_LOOKAHEAD = 30`
`/opt/splunk/etc/apps/TA-wunderground/default/props.conf REPORT-extjson = wunder_ext_json`
`/opt/splunk/etc/apps/TA-wunderground/default/props.conf SHOULD_LINEMERGE = true`
`/opt/splunk/etc/apps/TA-wunderground/default/props.conf TIME_PREFIX = observation_epoch`
`/opt/splunk/etc/apps/TA-wunderground/default/props.conf TRUNCATE = 1000000`

Resources and Questions

- IRC #splunk on efnet.org (look for alacer)
- docs.splunk.com
- answers.splunk.com (I'm alacer cogitatus)
- wiki.splunk.com
- Slack! Join a User Group! (www.splunk402.com/chat)
- The Splunk Trust - We are here to help! (find us on answers by our fez icon!)

THANK YOU

.conf2016



splunk®