Splunk Introduction

COMP90073
Security Analytics

Dr. Yi Han, CIS

Semester 2, 2021

THE UNIVERSITY OF MELBOURNE

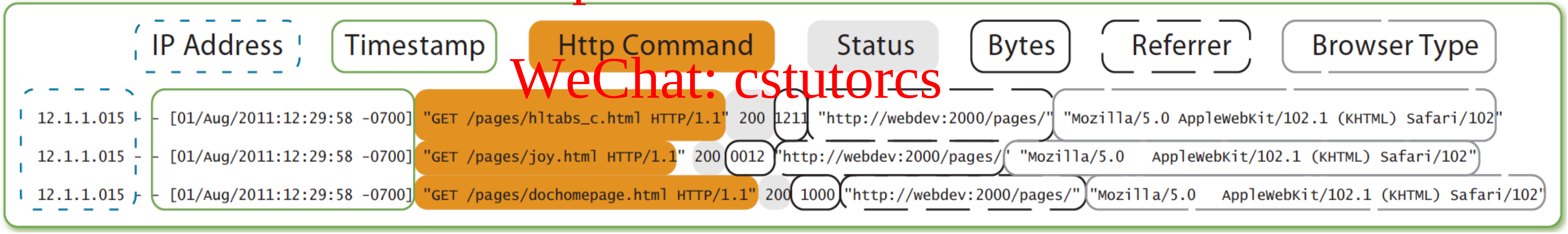
Outline

- What is Splunk & Why Splunk
- Splunk Software
- Search Processing Language (SPL)

What is Splunk & Why Splunk

A software for searching, monitoring, and analysing **machine generated big data** using a web-style interface

A typical web server log

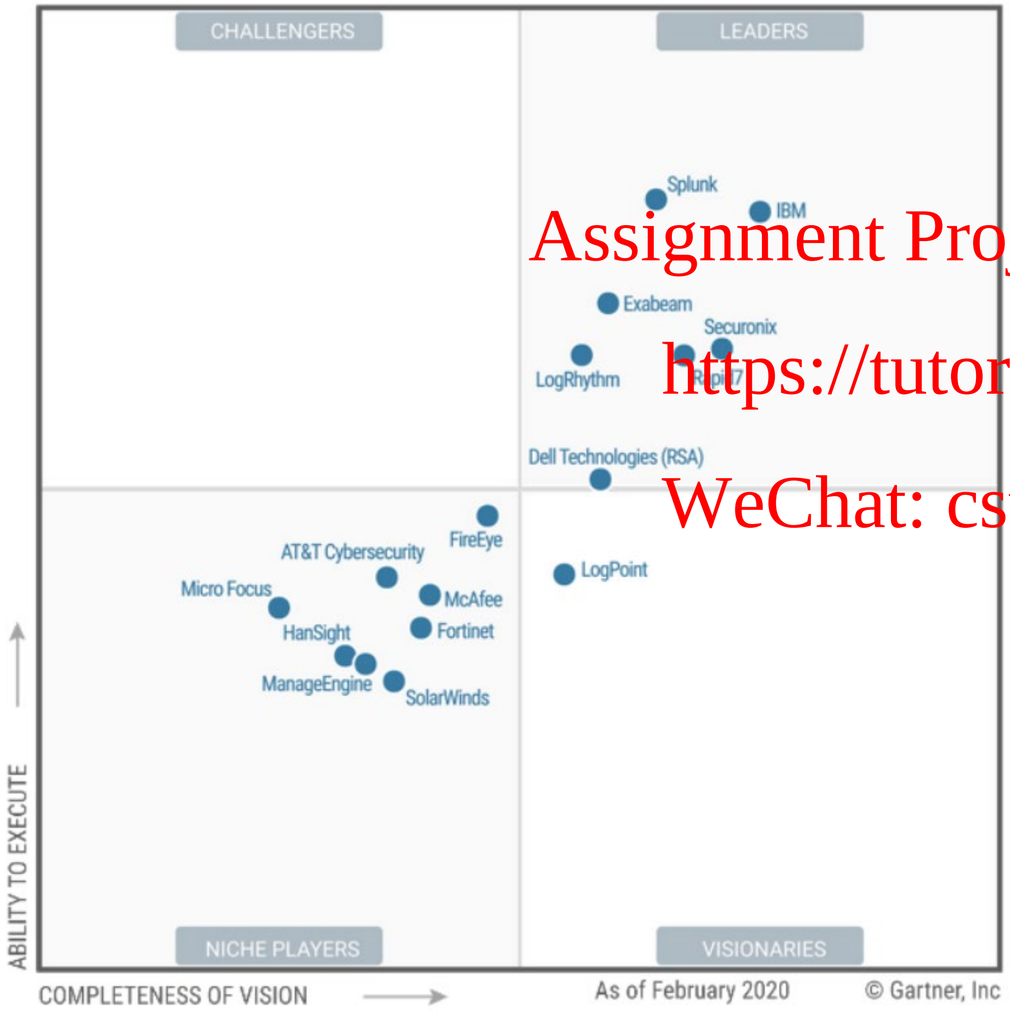


```
12.1.1.015 - - [01/Aug/2011:12:29:58 -0700] "GET /pages/hltabs_c.html HTTP/1.1" 200 1211 "http://webdev:2000/pages/" "Mozilla/5.0 AppleWebKit/102.1 (KHTML) Safari/102"
12.1.1.015 - - [01/Aug/2011:12:29:58 -0700] "GET /pages/joy.html HTTP/1.1" 200 0012 "http://webdev:2000/pages/" "Mozilla/5.0 AppleWebKit/102.1 (KHTML) Safari/102"
12.1.1.015 - - [01/Aug/2011:12:29:58 -0700] "GET /pages/dochomepage.html HTTP/1.1" 200 1000 "http://webdev:2000/pages/" "Mozilla/5.0 AppleWebKit/102.1 (KHTML) Safari/102"
```

Challenging to analyse multiple logs in real-time to detect security events!

What is Splunk & Why Splunk

Gartner 2020 Magic Quadrant for Security Information and Event Management (SIEM)



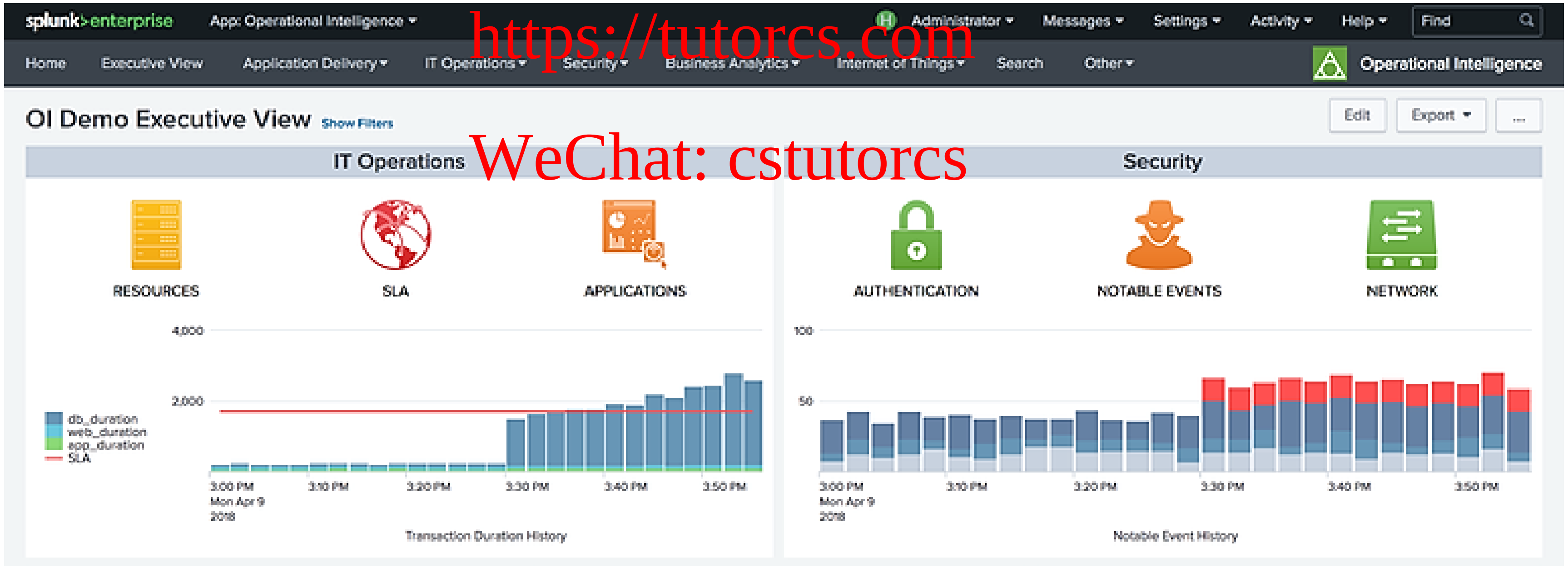
- Advanced threat detection and response solution
  - User and entity behavior analytics (UEBA)
  - Endpoint detection and response (EDR)
  - Automated threat intelligence
  - Real-time dashboards and reports
  - And more …

Splunk Software

- Splunk Capabilities
- Splunk Architecture
- What Can be Indexed
- Web Interface Overview
- Search & Reporting
- Events & Fields
- Default Fields
- Data Type & Common Operators

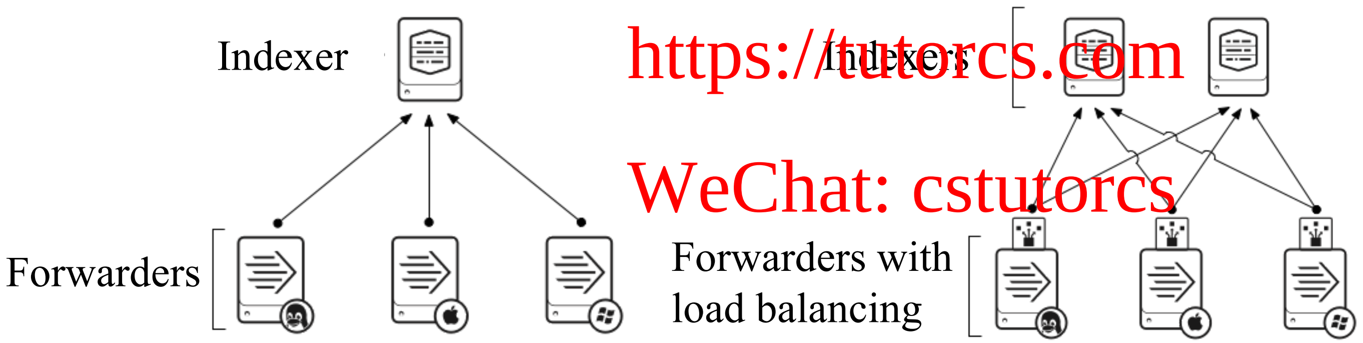
Splunk Capabilities

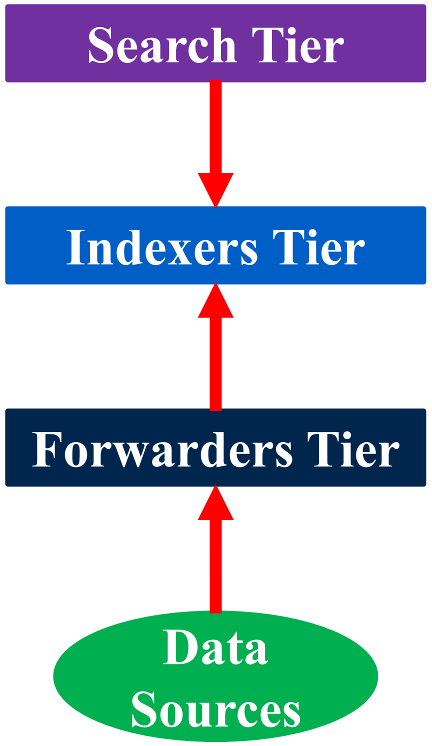
- Collect, index, and correlate machine data in **real-time**
  - **Indexing:** transforming data into a series of *events* that contain searchable ***fields*** *(e.g. IP addresses of source and destination in a network packet)*
    - Index: A repository for Splunk data
- Generate graphs, reports, alerts, dashboards and visualizations



Splunk Architecture

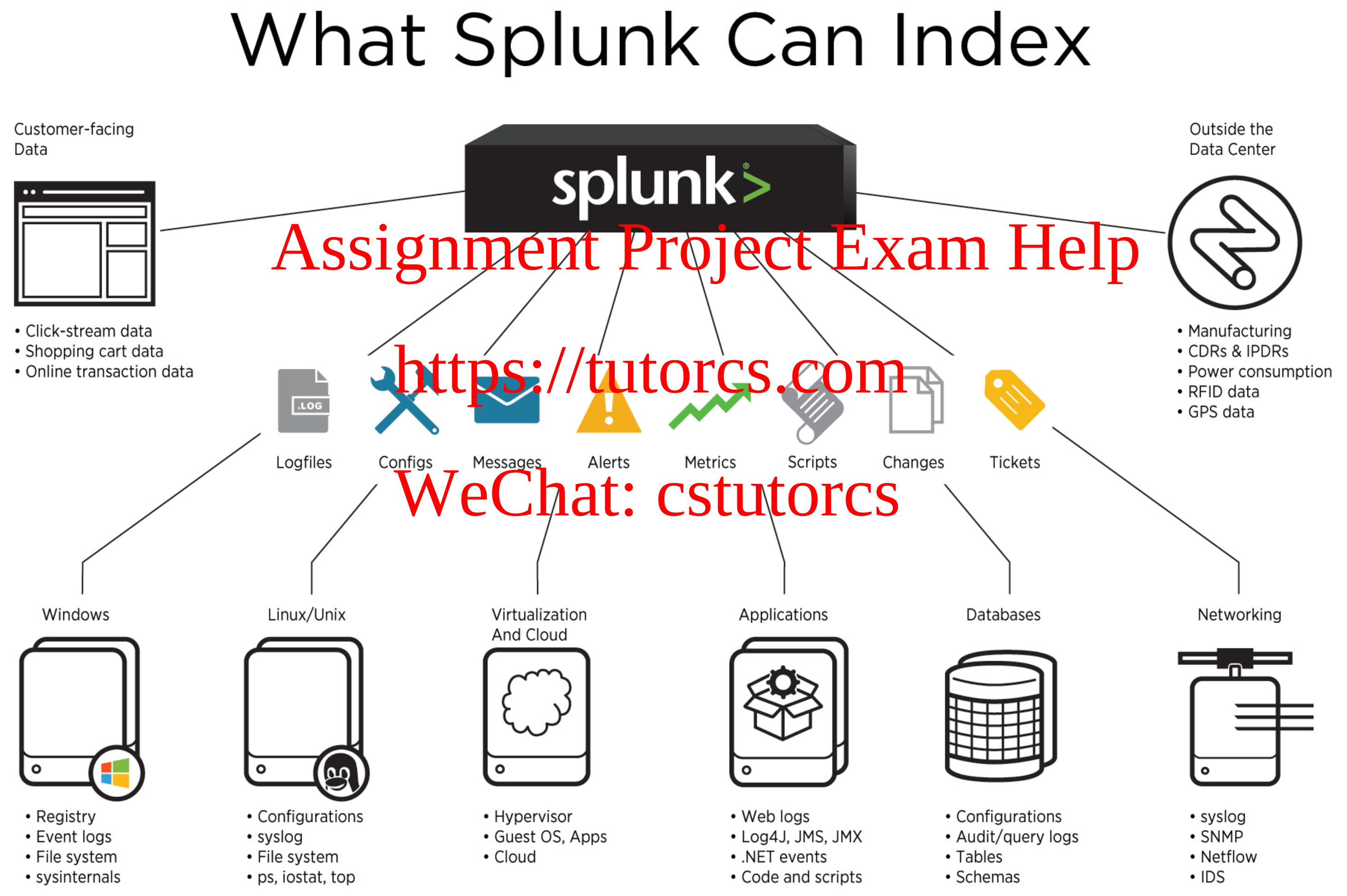
- Data sources: logs, file systems, Netflow, etc.
- Splunk forwarders: forwards the data from different data input sources to the indexers
- Splunk indexers: creates and manages indexes for the incoming data



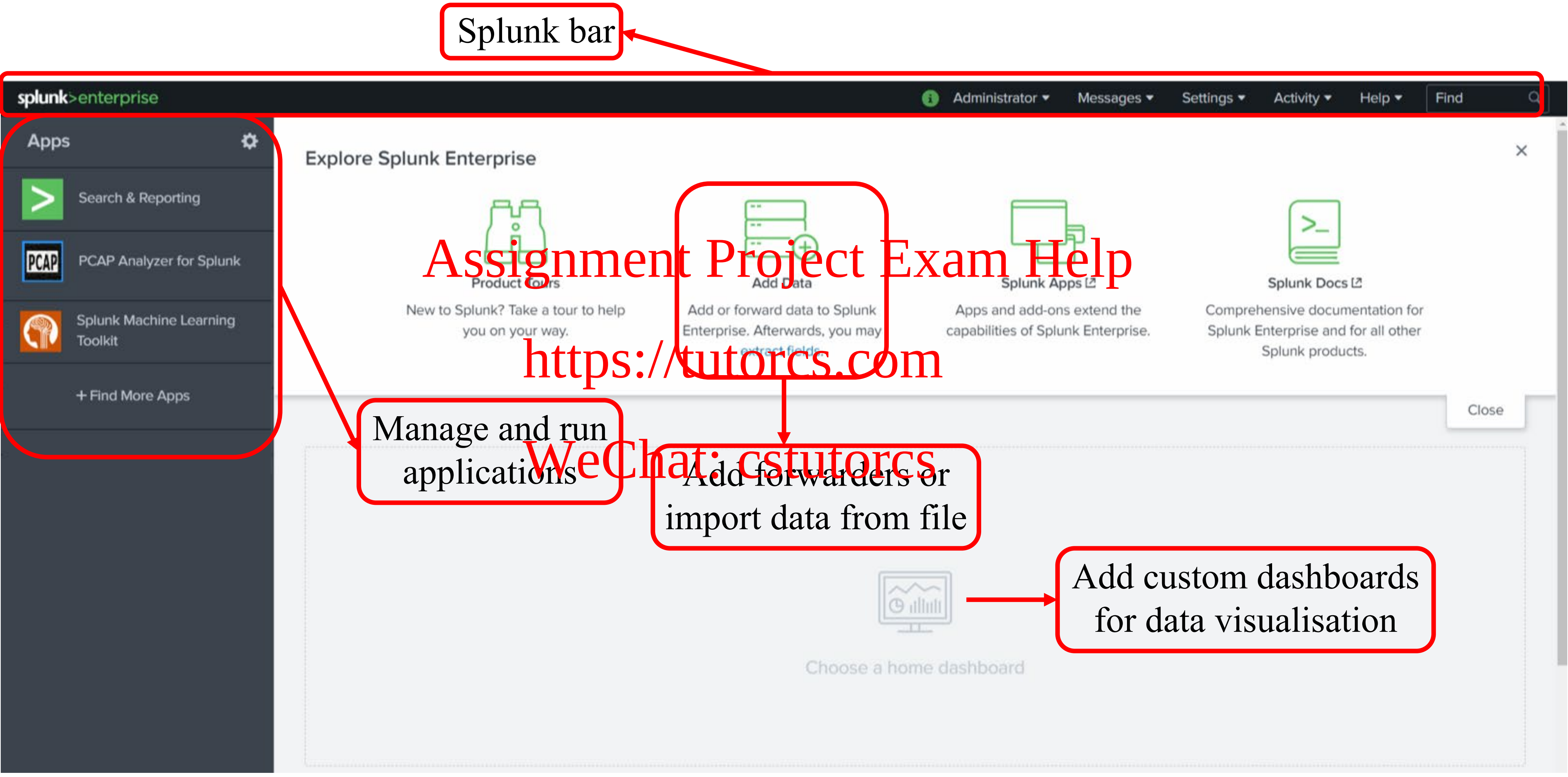


- Splunk search tier: includes search heads that process the search queries from users on the indexed data

What Can be Indexed



Web Interface Overview



New to Splunk? Take a tour to help you on your way.

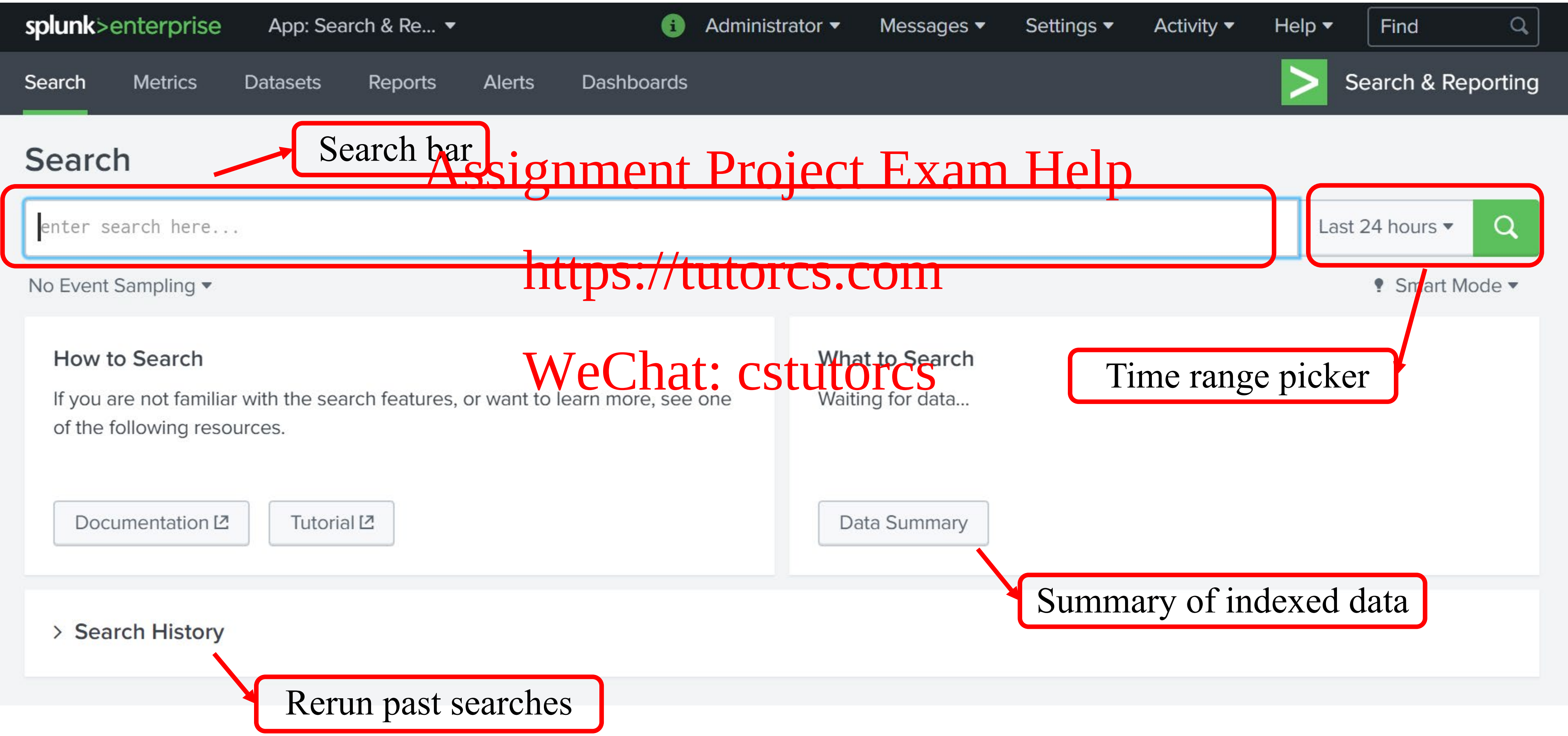
Add or forward data to Splunk Enterprise. Afterwards, you may extract fields.

Apps and add-ons extend the capabilities of Splunk Enterprise.

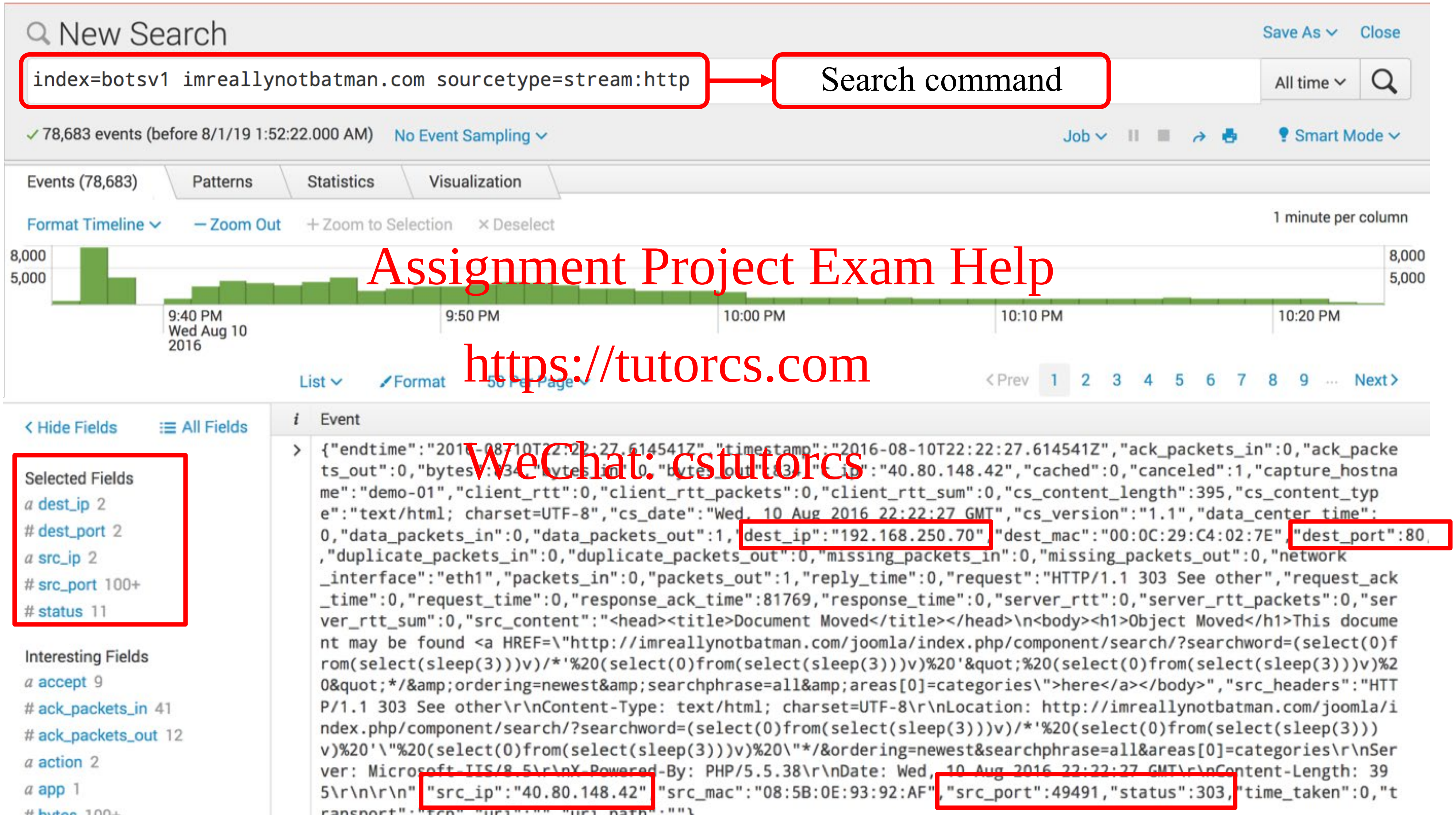
Comprehensive documentation for Splunk Enterprise and for all other Splunk products.

THE UNIVERSITY OF MELBOURNE

Search & Reporting



Event & Fields



```
index=botsv1 imreallynotbatman.com sourcetype=stream:http
```

✓78,683 events (before 8/1/19 1:52:22.000 AM)

```
{"endtime":"2016-08-10T22:22:27.614541Z","timestamp":"2016-08-10T22:22:27.614541Z","ack_packe
ts_out":0,"bytes":834,"bytes_in":0,"bytes_out":834,"c_ip":"40.80.148.42","cached":0,"canceled":1,"capture_hostna
me":"demo-01","client_rtt":0,"client_rtt_packets":0,"client_rtt_sum":0,"cs_content_length":395,"cs_content_typ
e":"text/html; charset=UTF-8","cs_date":"Wed, 10 Aug 2016 22:22:27 GMT","cs_version":"1.1","data_center_time":
0,"data_packets_in":0,"data_packets_out":1,"dest_ip":"192.168.250.70","dest_mac":"00:0C:29:C4:02:7E","dest_port":80
,"duplicate_packets_in":0,"duplicate_packets_out":0,"missing_packets_in":0,"missing_packets_out":0,"network
_interface":"eth1","packets_in":0,"packets_out":1,"reply_time":0,"request":"HTTP/1.1 303 See other","request_ack
_time":0,"request_time":0,"response_ack_time":81769,"response_time":0,"server_rtt":0,"server_rtt_packets":0,"ser
ver_rtt_sum":0,"src_content":"<head><title>Document Moved</title></head>\n<body><h1>Object Moved</h1>This docume
nt may be found <a HREF=\"http://imreallynotbatman.com/joomla/index.php/component/search/?searchword=(select(0)f
rom(select(sleep(3)))v)/*'%20(select(0)from(select(sleep(3)))v)%20'&quot;%20(select(0)from(select(sleep(3)))v)%2
0&quot;*/&amp;ordering=newest&amp;searchphrase=all&amp;areas[0]=categories\">here</a></body>","src_headers":"HTT
P/1.1 303 See other\r\nContent-Type: text/html; charset=UTF-8\r\nLocation: http://imreallynotbatman.com/joomla/i
ndex.php/component/search/?searchword=(select(0)from(select(sleep(3)))v)/*'%20(select(0)from(select(sleep(3)))
v)%20'\"%20(select(0)from(select(sleep(3)))v)%20\"*/&ordering=newest&searchphrase=all&areas[0]=categories\r\nSer
ver: Microsoft-IIS/8.5\r\nX-Powered-By: PHP/5.5.38\r\nDate: Wed, 10 Aug 2016 22:22:27 GMT\r\nContent-Length: 39
5\r\n\r\n","src_ip":"40.80.148.42","src_mac":"08:5B:0E:93:92:AF","src_port":49491,"status":303,"time_taken":0,"t
ransport":"tcp","uri":"","uri_path":""}
```

Data Source: https://live.splunk.com/splunk-security-dataset-project

Default Fields

- Shell scripts, python scripts, Windows batch files, PowerShell, etc., can be used to customise the data indexing and generate useful fields
- There are several internal and default fields that are automatically generated by Splunk

Type of field	List of fields	Description
Internal fields: Contain general information about events	_raw	Original raw data of an event
	_time	An event's timestamp expressed in Unix time
	_indextime	The time that an event was indexed
	_cd	An address for an event within the index
	_bkt	The bucket that an event is stored in

Default Fields

Type of field	List of fields	Description
Default fields: Contain information about where an event originated	host	Hostname/IP address of the device that generated the event (e.g., cisco_router)
	index	The name of the index in which a given event is indexed (e.g., default is "main")
	linecount	The number of lines an event contains
	punct	The punctuation pattern that is extracted from an event
	source	The file, stream, or other input from which an event originates (e.g., stream:http)
	sourcetype	The format of the data input from which the event originates (e.g. syslog)
	splunk_server	The Splunk server containing the event
	timestamp	An event's timestamp value

Default Fields

Type of field	List of fields	Description
Default datetime fields: Contain additional searchable granularity to event timestamps	date_hour	The hour in which an event occurred
	date_mday	The day of the month on which an event occurred
	date_minute	The minute in which an event occurred
	date_month	The month in which an event occurred
	date_second	The seconds portion of an event's timestamp
	date_wday	The day of the week on which an event occurred
	date_year	The year in which an event occurred
	date_zone	The value of time for the local time-zone of an event

Data Types & Common Operators

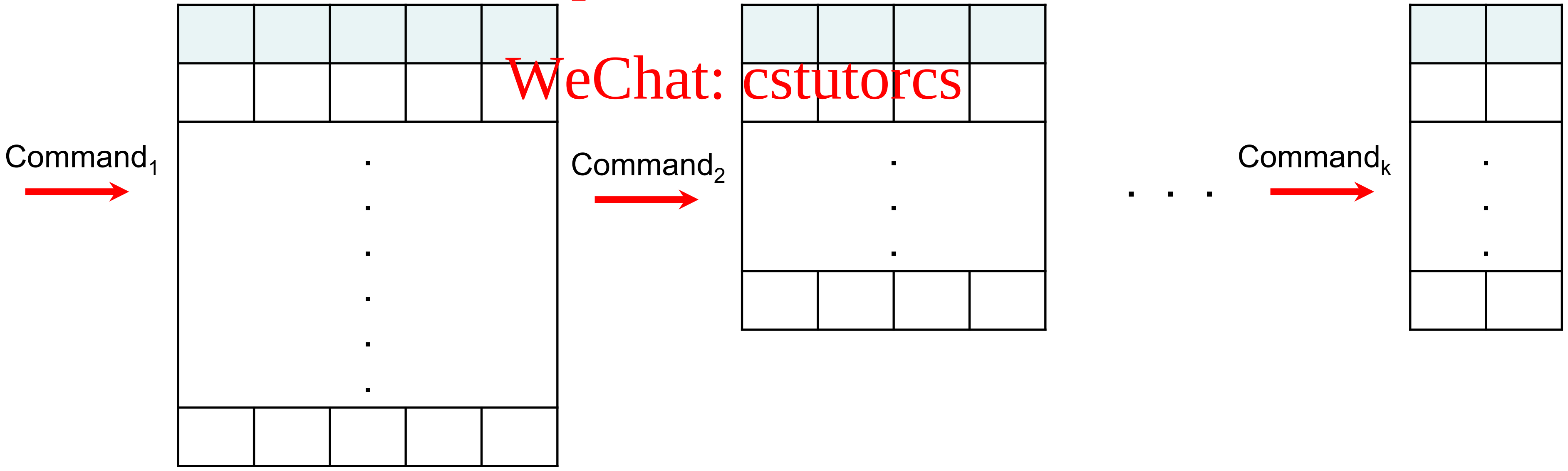
- Data types: bool, int, float, string
- Comparison operators: = != < <= > >=
- Logical operators: AND, OR, NOT
  - Clause “src_port !=80” is different from “NOT src_port=80”
    - Records with missing value of “src_port” field are returned in the second clause but are not returned in the first one
  - If no logical operator is used between clauses, the default operator is AND
    - “src_port !=80 host=server01” is equivalent to “src_port !=80 AND host=server01”

Search Processing Language (SPL)

- Filtering Results
- Sorting & Grouping Results
- Filtering & Modifying Fields

Common SPL Commands - Pipe

- Common search string in SPL: command_1 | command_2 | … | command_k
- Results after the pipe character "|" are used as input for its following command
- The pipe character is always followed by an SPL command



Common SPL Commands

- “search” command is implicitly applied in the beginning of the search pipeline and you should not use it explicitly in this location
  - Example: “src_port=80 | top dest_ip”

$\xrightarrow{}$ “search” command is implicitly applied here

Category	Description	Commands
Filtering Results	Taking a set of results and filtering them into a smaller set of results	**search**, **where**, dedup, **head**, **tail**
Sorting Results	Ordering (and optionally limiting the number of) results	**sort**
Grouping Results	Grouping events for identifying patterns	**transaction**
Reporting Results	Generating a summary of results for reporting	**top/rare, table, stats, chart**, timechart
Filtering, Modifying, and Adding Fields	Filtering out some fields to focus on most related ones, modifying or adding fields to enrich results	**fields, replace, rename, eval, rex**, lookup

Source: https://docs.splunk.com/

Common SPL Commands – Syntax Tips

- Required arguments are shown in angle brackets < >
- Optional arguments are enclosed in square brackets [ ]
- Group arguments are shown in parenthesis ()
- Repeating arguments are shown by ellipsis …
- Example
  - Syntax: replace (<string1> WITH <string2>)... [IN <field-list>]
  - Example: replace 200 WITH OK 404 WITH "Not Found" IN $\boxed{\text{status}}$

HTTP status field
in indexed data

THE UNIVERSITY OF MELBOURNE

Filtering the Results

Search command

- Filters events from Splunk indexes given a set of queried conditions
- Syntax: search <logical-expression> [AND/OR/NOT <logical-expression>]
- logical-expression
  - comparison-expression
  - index-expression
  - time-opts → You can also use the time range picker for time options
- Precedence of logical operators in search command: expressions with parenthesis, then NOT then OR then AND

THE UNIVERSITY OF MELBOURNE

Search command: comparison-expression

- <field><comparison-operator><value>
  - Examples: src_port < 100, src_ip=192.168.10.1
- <field> IN (<value-list>)
  - Example: dest_port IN (21,80,8080)
  - IN operator checks if a value is a member of a group of values
- Search command examples for the toy HTTP data:
  - search status >= 400
    - Returns events with error in HTTP requests
  - search status IN (401,403)
    - Returns events with unauthorized or Forbidden HTTP requests

Search command: index-expression

- "<string>"
  - Keywords or quoted phrases to match, Examples: fail*, login, "http://"
    - Wildcard: asterisk wildcard (*) character is used to match an unrestricted number of characters in a string
- <search-modifier>
  - <sourcetype-specifier> | <host-specifier> | <source-specifier> | <splunk_server-specifier>, etc.
  - Example: sourcetype=syslog
- Search example:
  - search sourcetype=stream:http fail* password
    - This is equivalent to "search sourcetype=stream:http AND fail* AND password"

Search command: time-opts

- [<timeformat>] (<time-modifier>)...
- timeformat
  - timeformat=…
  - Example: timeformat=%d/%m/%Y:%H:%M:%S
  - Default time format is %m/%d/%Y:%H:%M:%S
- <time-modifier> can be exact time or relative time
  - earliest, latest, _index_earliest, _index_latest, now(), time()
  - [±]<time_integer><time_unit>@<time_unit>
  - Example: "earliest=-3d@d latest=now()"
- Hint: you can use the web interface for setting the time options

Time unit	second	minute	hour	day	week	month	quarter	year
Valid unit abbreviations	s, sec, secs, second, seconds	m, min, minute, minutes	h, hr, hrs, hour, hours	d, day, days	w, week, weeks	mon, month, months	q, qtr, qtrs, quarter, quarters	y, yr, yrs, year, years

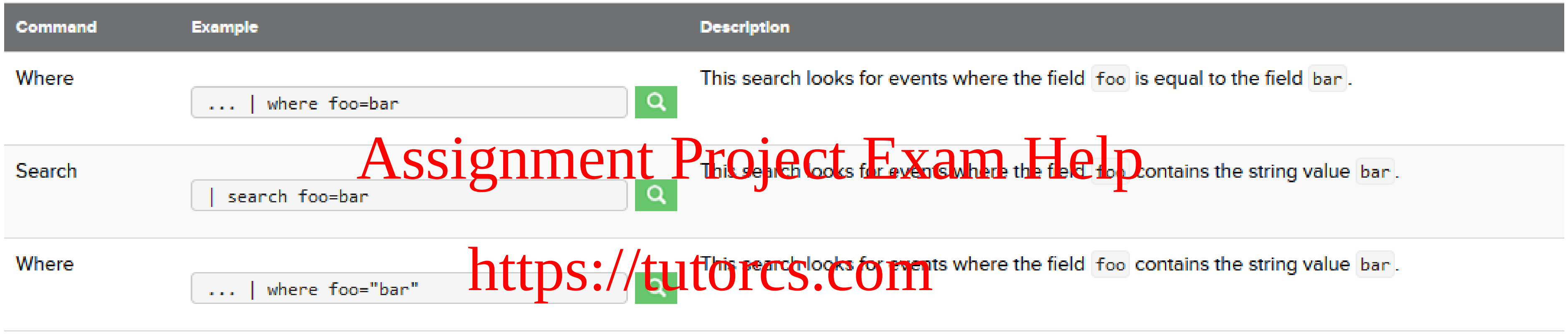
THE UNIVERSITY OF MELBOURNE

Tips for search command

- Field names are by default case-sensitive
- Literals are not case sensitive by default
  - Example: searching for login, Login, or "Login" all return same results
  - Use CASE(<string>) for case-sensitive search of the field values
    - CASE(Login) only returns events that include Login (not login)
- Splunk searches for whole word
  - Search results for “fail” and “failure” ☛ use asterisk wildcard (*) ☛ fail*
- For phrases or field values containing breaking characters, e.g., whitespace, commas, pipes, square brackets and equal sign use quotation marks
  - Examples: host=“server 1”
  - Use backslash (\) to scape quote in the filed value, e.g., host=“server\” 1” → looking for records with host name equal to <server” 1>

Where command

- Quoted strings are interpreted as literals
- Unquoted strings are treated as a field name → Compare two different fields

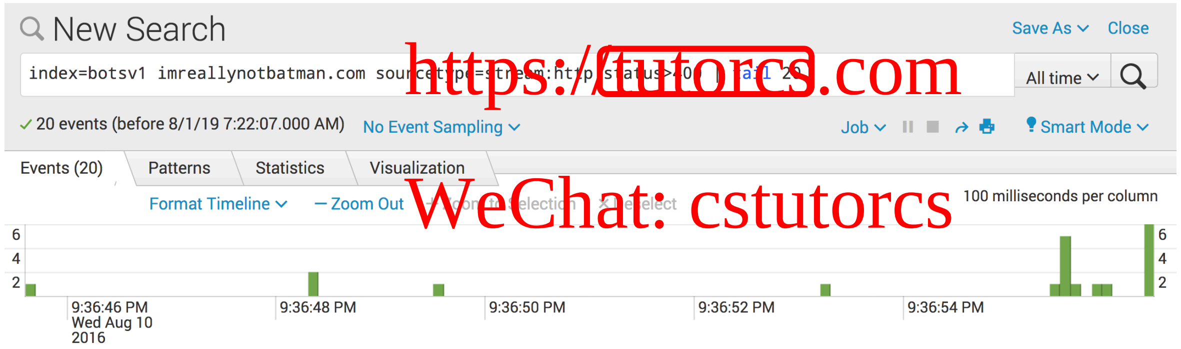


Command	Example	Description
Where	`... | where foo=bar`	This search looks for events where the field `foo` is equal to the field `bar`.
Search	`| search foo=bar`	This search looks for events where the field `foo` contains the string value `bar`.
Where	`... | where foo="bar"`	This search looks for events where the field `foo` contains the string value `bar`.

- Can also be used with IN operator and a value-list
  - Example: … | where dest_port IN (80,8080)
- Precedence of logical operators in where: expressions with parenthesis, then NOT then AND then OR
- Examples
  - … | where src_port=dst_port
  - … | where bytes_in>2*bytes_out

Head and tail commands

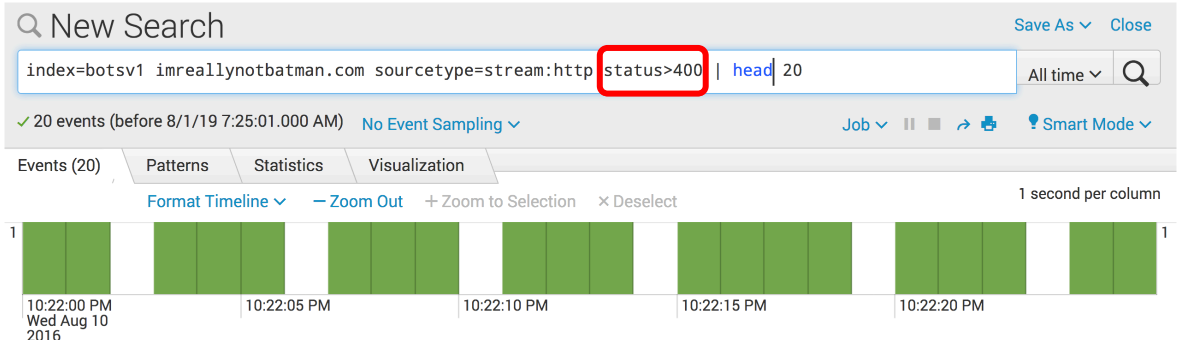
- Head returns the most recent results of a search
  - … | head 25
- Tail returns the earliest results of a search
  - ... | tail 15
- If the integer argument is not given, both commands return 10 results by default



```
index=botsv1 imreallynotbatman.com sourcetype=stream:http status>400 | tail 20
```

✓ 20 events (before 8/1/19 7:22:07.000 AM)

status>400 | tail 20



```
index=botsv1 imreallynotbatman.com sourcetype=stream:http status>400 | head 20
```

✓ 20 events (before 8/1/19 7:25:01.000 AM)

status>400 | head 20

BOTS: https://live.splunk.com/splunk-security-dataset-project

THE UNIVERSITY OF MELBOURNE

Sorting & Grouping Results

Sort command

- To change the ordering/number of the results
- Syntax: sort [<count>] <sort-by-clause>... [desc]
- Default value of the optional field count is 10,000; pass 0 to return all the results
- sort-by-clause: [±] <sort-field>[, [±] <sort-field>]
  - The value of sort-filed can be a field (such as "src_port") or
    - auto(<field>) → Splunk chooses the type of field for sorting
    - ip(<field>) → Splunk treats the field values as IP address for sorting
    - num(<field>) → Splunk treats the field values as number for sorting
    - str(<field>) → Splunk treats the field values as string for sorting
- Default sorting order is ascending
  - Use minus sign for descending order, e.g., sort –src_port, +ip(src_ip)
- Examples:
  - ... | sort lastname, -firstname
  - ... | sort 100 -num(size), +str(source)

Transaction command

- Group of conceptually-related events that spans time
  - Examples
    - Different events from the same source and the same host
    - Different events from different sources but from the same host
    - Similar events from different hosts and different sources
    - A set of events related to a firewall intrusion incident
- Syntax: transaction [<field-list>] [name=<transaction-name>] [<transaction_definition-options>...]

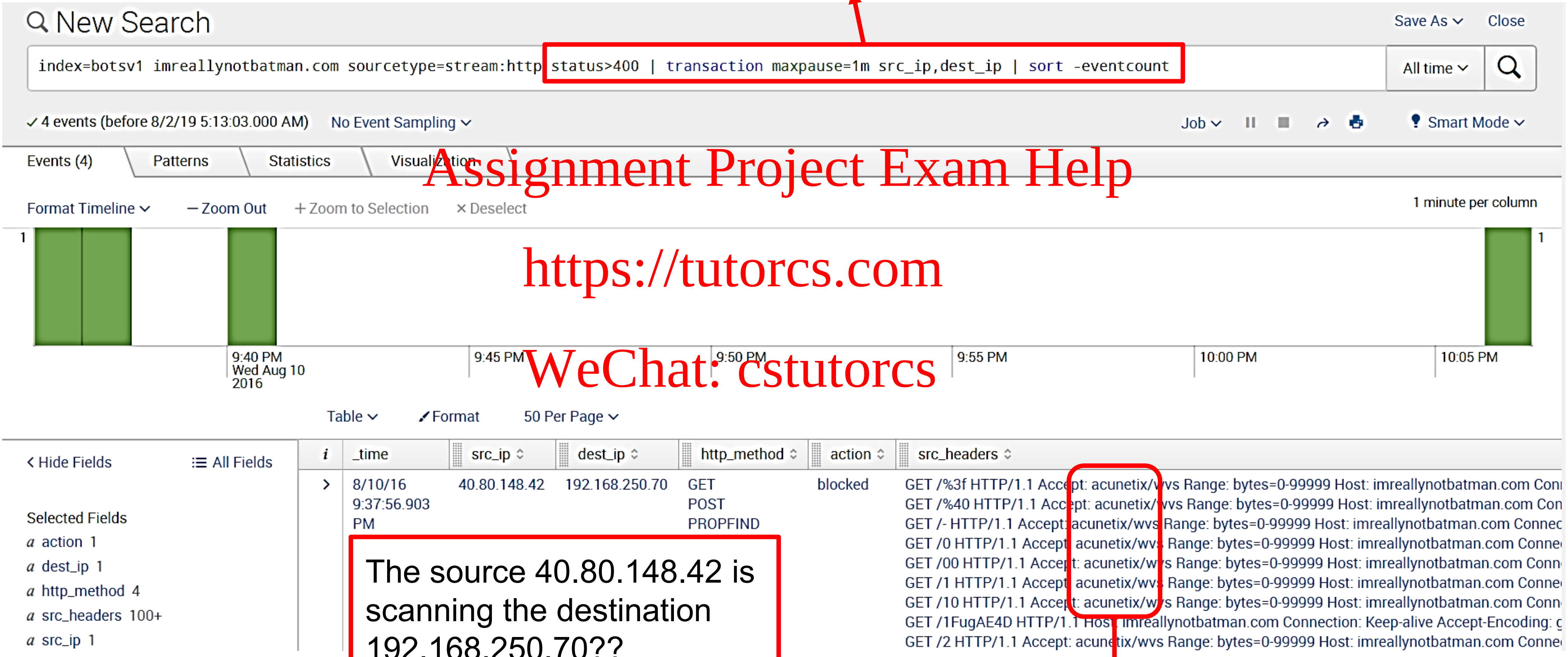
- This command adds two fields to the raw events: *duration* and *eventcount*
- The argument field-list specifies one field or more field names to group events into transactions based on the values of the field(s)
  - The relationship among the fields can be conjunction, disjunction, transitive, …

Transaction command: transaction definition options

- transaction-definition-options
  - endswith=<filter-string>, startswith=<filter-string>:
    - To start or end a transaction if the filter-string is satisfied by an event
  - maxspan=<int>time-unit → time-unit options s, m, h, d
    - Events in the transaction must span less than integer specified for maxspan. Events that exceed the maxspan limit are treated as part of a separate transaction
  - maxpause=<int> time-unit
    - To specify the maximum length of time for the pause between the events in a transaction
  - maxevents=<int>
    - To specify the maximum number of events in a transaction. The default value is1000.
  - A negative value for each of these constraints means that there is no limit on the its value

Transaction command: example

status>400 | transaction maxpause=1m src_ip,dest_ip | sort -eventcount



```
index=botsv1 imreallynotbatman.com sourcetype=stream:http status>400 | transaction maxpause=1m src_ip,dest_ip | sort -eventcount
```

✓ 4 events (before 8/2/19 5:13:03.000 AM)

i	_time	src_ip	dest_ip	http_method	action	src_headers
>	8/10/16 9:37:56.903 PM	40.80.148.42	192.168.250.70	GET POST PROPFIND	blocked	GET /%3f HTTP/1.1 Accept: acunetix/wvs Range: bytes=0-99999 Host: imreallynotbatman.com Con GET /%40 HTTP/1.1 Accept: acunetix/wvs Range: bytes=0-99999 Host: imreallynotbatman.com Con GET /- HTTP/1.1 Accept: acunetix/wvs Range: bytes=0-99999 Host: imreallynotbatman.com Connec GET /0 HTTP/1.1 Accept: acunetix/wvs Range: bytes=0-99999 Host: imreallynotbatman.com Conne GET /00 HTTP/1.1 Accept: acunetix/wvs Range: bytes=0-99999 Host: imreallynotbatman.com Conn GET /1 HTTP/1.1 Accept: acunetix/wvs Range: bytes=0-99999 Host: imreallynotbatman.com Conne GET /10 HTTP/1.1 Accept: acunetix/wvs Range: bytes=0-99999 Host: imreallynotbatman.com Conn GET /1FugAE4D HTTP/1.1 Host: imreallynotbatman.com Connection: Keep-alive Accept-Encoding: g GET /2 HTTP/1.1 Accept: acunetix/wvs Range: bytes=0-99999 Host: imreallynotbatman.com Conne

Selected Fields
- a action 1
- a dest_ip 1
- a http_method 4
- a src_headers 100+
- a src_ip 1

The source 40.80.148.42 is scanning the destination 192.168.250.70??

Acunetix is a vulnerability scanner

BOTS: https://live.splunk.com/splunk-security-dataset-project

THE UNIVERSITY OF MELBOURNE

Reporting Results

Commands for statistical calculations

- Calculate aggregate statistics (average, count, sum, …) over a results set
- Commands
  - **stats**: returns a table of results where each row represents a single unique combination of the values grouped by a set of chosen fields
    - See others: eventstats, streamstats, geostats
  - chart: similar to stats but creates tabular data output suitable for charting
  - timechart: creates a chart for a statistical aggregation applied to a field against time as the x-axis

Stats command

Syntax: stats [partitions=<num>] [allnum=<bool>] [delim=<string>]
(**<stats-agg-term>... or <sparkline-agg-term>...**) [<by-clause>]

Lower-case "or" in these slides is used to show alternative available options

- stats-agg-term: <stats-func>(<field>) [AS <field>]
  - Choices of stats-func → next slide
  - Input field argument can be an existing field-name (e.g., src_port) or evaled-field created using eval command inside stats
    - stats count(eval(src_port=80)) → evaled-field is "eval(src_port=80)"
  - Wildcard field names can be used: this option returns separate results applying stats-func on each field: stats count(eval(*_port=80))
  - The optional argument [AS <field>] can be used to rename the output fields and can be wildcard field names:
    - Example 1: "stats count(eval(*_port=80)) AS *_port80"
- <by-clause>: Split output based on a set of given fields. If omitted, the stats is computed for the entire input result set. Example: stats distinct_count(src_port) BY src_ip

Options for stats-func

Type of function	Supported functions and syntax			

Aggregate functions	avg()
	count()
	distinct_count()
	estdc()
	estdc_error()
		exactperc<int>()
		max()
		median()
		min()
		mode()
			perc<int>()
			range()
			stdev()
			stdevp()
				sum()
				sumsq()
				upperperc<int>()
				var()
				varp()

Event order functions	first()	last()		
Multi-value stats and chart functions	list()	values()		

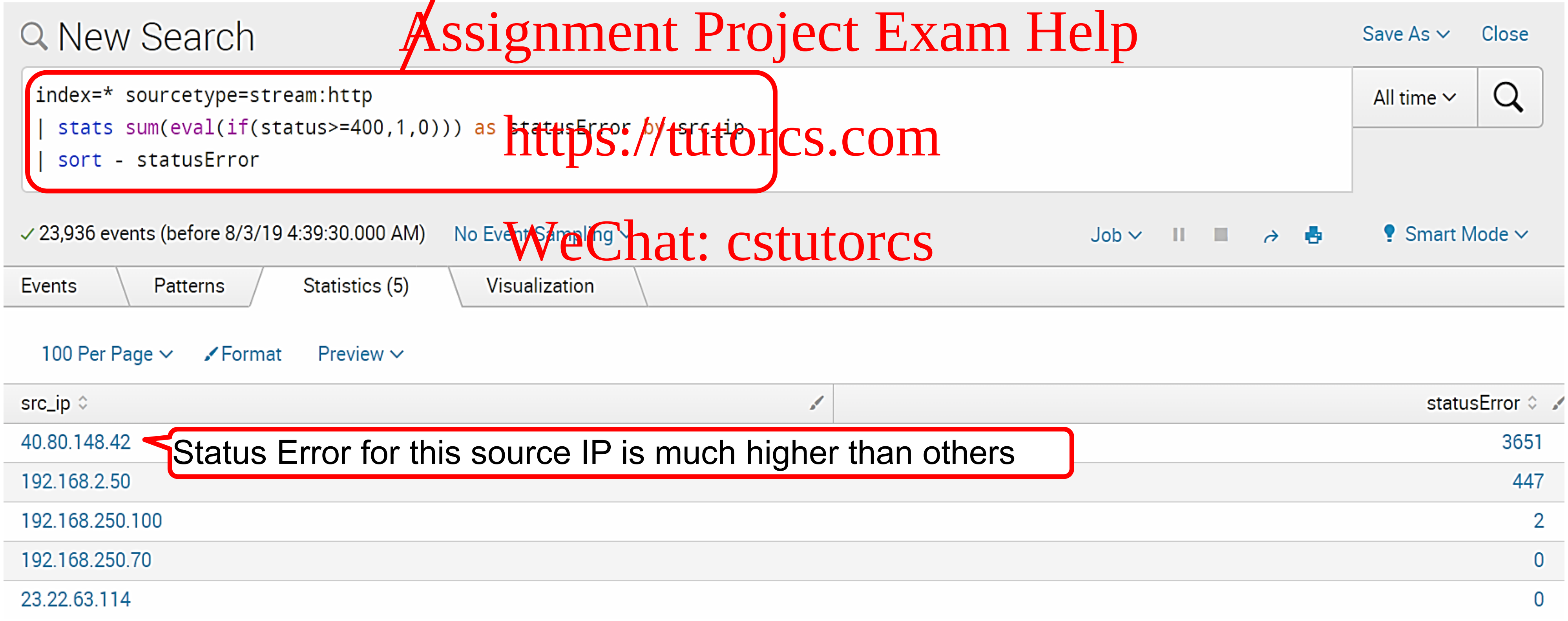
Time functions	earliest()
	earliest_time()
		latest()
		latest_time()
			rate()	

More detail on the functions:
https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/CommonStatsFunctions

Stats command (example)

… | stats sum(eval(if(status>=400,1,0))) AS statusError BY src_ip | sort - statusError

Execution per src_ip:
1. eval(if(status>=400,1,0)) → 0, 1
2. stats command sums over the output of eval **splitting by source IP address**
3. sort command sorts the results



```
index=* sourcetype=stream:http
| stats sum(eval(if(status>=400,1,0))) as statusError by src_ip
| sort - statusError
```

✓ 23,936 events (before 8/3/19 4:39:30.000 AM)

src_ip	statusError
40.80.148.42	3651
192.168.2.50	447
192.168.250.100	2
192.168.250.70	0
23.22.63.114	0

Status Error for this source IP is much higher than others

BOTS: https://live.splunk.com/splunk-security-dataset-project

THE UNIVERSITY OF MELBOURNE

Stats command (example)

Scenario

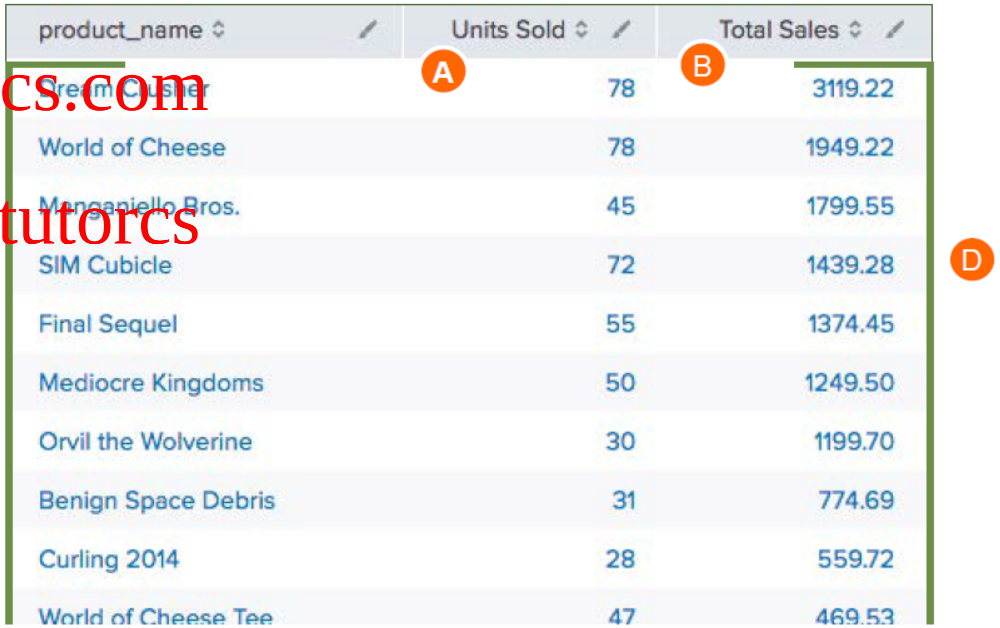
Report the number of retail units sold and sales revenue for each product during the previous week.

```
index=sales sourcetype=vendor_sales
| stats A count(price) as "Units Sold"
B sum(price) as "Total Sales"   by product_name C
| sort -"Total Sales" D
```

A B A single stats command can have multiple functions

C The by clause is applied to both functions

D sort Total Sales in descending order

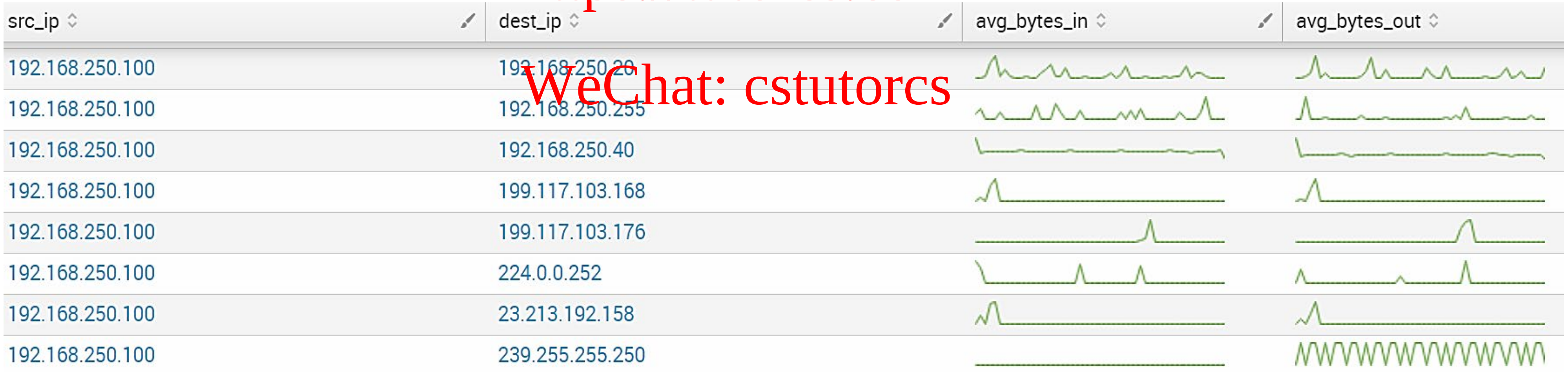


product_name	Units Sold	Total Sales
Cream Crusher	78	3119.22
World of Cheese	78	1949.22
Manganiello Bros.	45	1799.55
SIM Cubicle	72	1439.28
Final Sequel	55	1374.45
Mediocre Kingdoms	50	1249.50
Orvil the Wolverine	30	1199.70
Benign Space Debris	31	774.69
Curling 2014	28	559.72
World of Cheese Tee	47	469.53

Stats command: sparkline-agg-term

- Sparkline: an inline chart that appears within table cells in search results to display time-based trends associated with the primary key of each row
- Syntax: sparkline (<sparkline-func>(<wc-field>), <span-length>)
  - sparkline-func options: count(), mean(), avg(), stdev(), min(), max(), etc.
  - span-length examples: 1d, 10min, 1mon

Example: index=* | stats sparkline(avg(bytes_*),1m) AS avg_bytes_* BY src_ip,dest_ip



src_ip	dest_ip	avg_bytes_in	avg_bytes_out
192.168.250.100	192.168.250.20		
192.168.250.100	192.168.250.255		
192.168.250.100	192.168.250.40		
192.168.250.100	199.117.103.168		
192.168.250.100	199.117.103.176		
192.168.250.100	224.0.0.252		
192.168.250.100	23.213.192.158		
192.168.250.100	239.255.255.250		

These lines change as the search proceeds

BOTS: https://live.splunk.com/splunk-security-dataset-project

THE UNIVERSITY OF MELBOURNE

Stats command: other arguments

- partitions=<num>: partition the input for multithreaded computation
- allnum=<bool>: If true, numerical statistics is computed for a field if and only if all of the values of that field are numerical
- delim=<string>: if list() or values() statistical functions are used, specifies how the values in the aggregation are delimited. Default is space

THE UNIVERSITY OF MELBOURNE

Chart command

Syntax: chart (<stats-agg-term> or <sparkline-agg-term> or "("<eval-expression>")")...

[(BY <row-split> <column-split>) or [OVER <row-split>] [BY <column-split>]]

- row-split
  - <field> [<bin-options>]
  - bin-options: bins, span, …
    - Examples: bins=5, span=1min, …
- column-split
  - <field> [<tc-options>]... [<where-clause>]
  - tc-options: <bin-options>, otherstr=<string>, …

Compare stats and chart commands

```
chart count(eval(src_port=80)) AS port80 OVER dest_port bins=10 BY dest_ip
```

dest_port	10.120.137.110	10.120.251.250	. . .	10.186.60.244	10.85.245.109	OTHER
0-10000	590	566	. . .	417	453	139639
10000-20000	25	17		7	14	3309
⋮	⋮	⋮	⋮	⋮	⋮	⋮
60000-70000	4	7	. . .	8	4	1378

```
stats count(eval(src_port=80)) AS port80 BY dest_port, dest_ip
```

dest_port	dest_ip	port80
80	10.168.80.39	171
80	10.122.27.216	161
80	10.122.68.227	161
80	10.120.137.110	159
. . .		

Top and rare commands

- top [<N>] [<options>...] <field-list> [BY <field-list>]
  - Most common (optionally N) values for the fields
  - Example: “top src_ip dest_ip”
- rare [<options>...] <field-list> [BY <field-list>]
  - Least common (optionally N) values for the fields
- Two fields are added to events when using top and rare: *count* and *percentage*
- Optional by_clause is for grouping and ordering the results using other fields

top src_ip dest_ip dest_port

src_ip	dest_ip	dest_port	count	percent
40.80.148.42	192.168.250.70	80	5931	0.816
23.22.63.114	192.168.250.70	80	1236	0.170
40.80.148.42	192.168.250.40	8000	100	0.014

top src_ip dest_ip by dest_port

dest_port	src_ip	dest_ip	count	percent
80	40.80.148.42	192.168.250.70	5931	0.828
80	23.22.63.114	192.168.250.70	1236	0.172
8000	40.80.148.42	192.168.250.40	100	100

Top and rare commands: options

- showcount=<bool> for choosing to show the count values or not
- countfield=<string> for choosing another name for the count field
- showperc=<bool> for choosing to show the percentage values or not
- percentfield=<string> for choosing another name for the percentage field
- limit=<int> for specifying the number of results returned (default 10)
- useother=<bool> for adding a row to the results for all the other values
- otherstr=<string> for choosing a label for the new row for other values when useother=true

Table command

- table <wc-field-list>
  - Example: … | table *_ip *_port

dest_ip	src_ip	dest_port	src_port
192.168.250.40	192.168.250.100	8089	49772
192.168.250.40	192.168.250.100		
8.8.8.8	192.168.250.40	53	53273
8.8.8.8	192.168.250.40	53	53273
8.8.8.8	192.168.250.40	53	42173
8.8.8.8	192.168.250.40	53	42173

THE UNIVERSITY OF MELBOURNE

Filtering, Modifying & Adding Fields

THE UNIVERSITY OF MELBOURNE

Eval command

- Calculates the value of a new field based on other fields, whether numerically, by concatenation, or through Boolean logic
- Syntax: eval <field>=<expression>["," <field>=<expression>]...

The double quotation sign means mandatory use of comma

- <expression> can be a mathematical, string, or Boolean expression
  - If the expression
    - refers to field names with non-alphanumeric characters, the name should be in single quotation marks (e.g., 'src_port')
    - refers to literal strings, they should be in double quotation marks
- The output is stored in <field>
  - If the field already exists, eval overwrites the corresponding field values
  - The returned field values by eval cannot be Boolean (tostring() function can be used to convert results to string)

THE UNIVERSITY OF MELBOURNE

Functions for eval expressions

Type of function	Supported functions and syntax		

Comparison and Conditional functions	case(X,"Y",...)
	cidrmatch("X",Y)
	coalesce(X,...)
	false()
	if(X,Y,Z)
		in(VALUE-LIST)
		like(TEXT, PATTERN)
		match(SUBJECT, "REGEX")
		null()
			nullif(X,Y)
			searchmatch(X)
			true()
			validate(X,Y,...)

Conversion functions	printf("format",arguments)	tonumber(NUMSTR,BASE)	tostring(X,Y)

Cryptographic functions	md5(X)
	sha1(X)
		sha256(X)	sha512(X)

Date and Time functions	now()
	relative_time(X,Y)
		strftime(X,Y)
		strptime(X,Y)
			time()

More detail on the functions:
https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Eval

Functions for eval expressions

Type of function	Supported functions and syntax		

Informational functions	isbool(X)
	isint(X)
	isnotnull(X)
		isnull(X)
		isnum(X)
			isstr(X)
			typeof(X)

Mathematical functions	abs(X)
	ceiling(X)
	exact(X)
	exp(X)
		floor(X)
		ln(X)
		log(X,Y)
		pi()
			pow(X,Y)
			round(X,Y)
			sigfig(X)
			sqrt(X)

Multi-value eval functions	commands(X)
	mvappend(X,...)
	mvcount(MVFIELD)
	mvdedup(X)
		mvfilter(X)
		mvfind(MVFIELD,"REGEX")
		mvindex(MVFIELD,STARTINDEX,ENDINDEX)
		mvjoin(MVFIELD,STR)
			mvrange(X,Y,Z)
			mvsort(X)
			mvzip(X,Y,"Z")
			split(X,"Y")

More detail on the functions:
https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Eval

Functions for eval expressions

Type of function	Supported functions and syntax		
Statistical eval functions	max(X,...)	min(X,...)	random()

Text functions	len(X)
	lower(X)
	ltrim(X,Y)
	replace(X,Y,Z)
		rtrim(X,Y)
		spath(X,Y)
		substr(X,Y,Z)
		trim(X,Y)
			upper(X)
			urldecode(X)

Trigonometry and Hyperbolic functions	acos(X)
	acosh(X)
	asin(X)
	asinh(X)
	atan(X)
		atan2(X,Y)
		atanh(X)
		cos(X)
		cosh(X)
		hypot(X,Y)
			sin(X)
			sinh(X)
			tan(X)
			tanh(X)

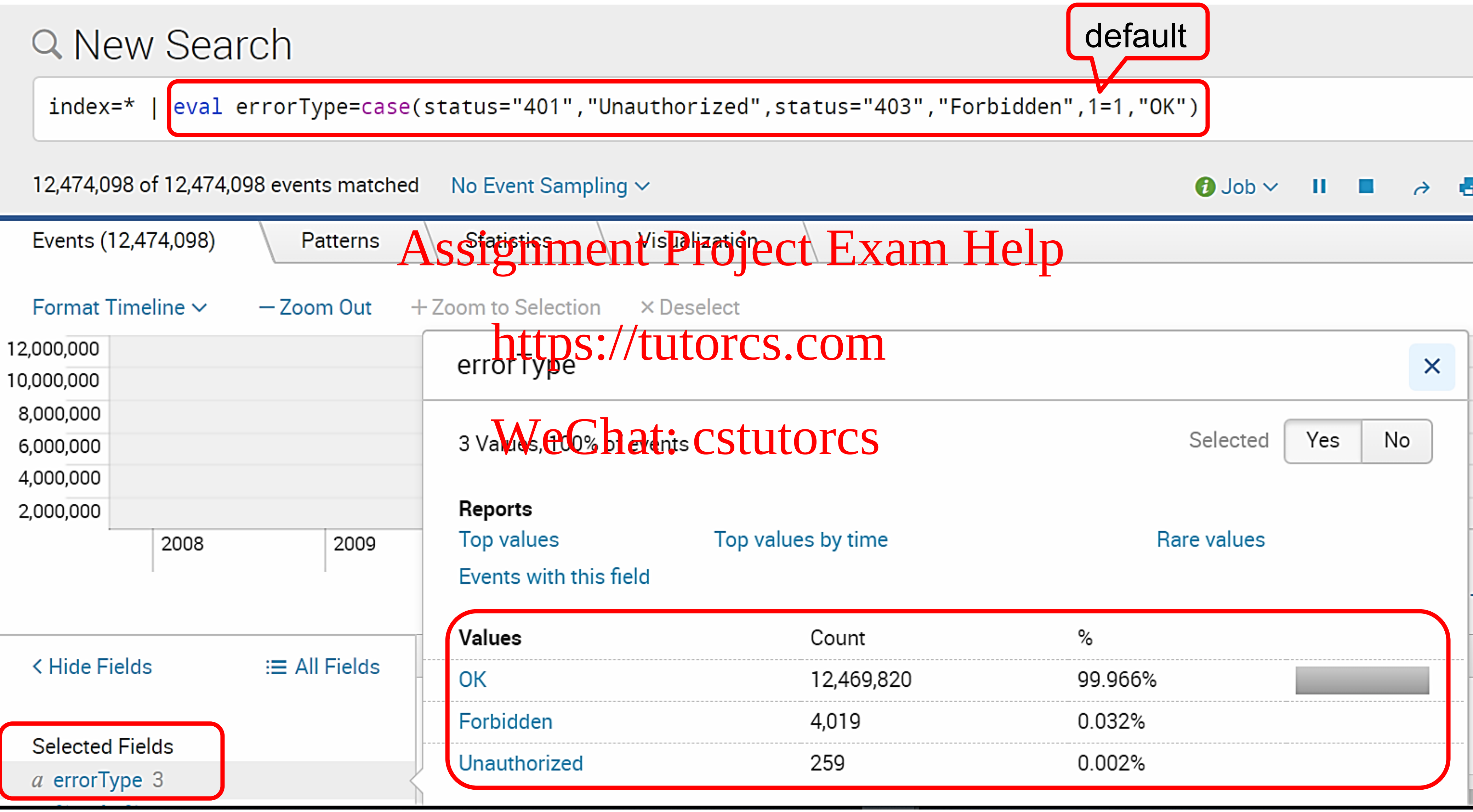
More detail on the functions:
https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Eval

THE UNIVERSITY OF MELBOURNE

Eval command examples

- Create a new field that contains the result of a calculation
  - ... | eval velocity=distance/time
- Use the if function to analyse field values
  - ... | eval error = if(status == 200, "OK", "Problem")
- Convert values to lowercase
  - ... | eval lowuser = lower(username)
- Calculate the sum of the areas of two circles
  - ... | eval sum_of_areas = pi() * pow(radius_a, 2) + pi() * pow(radius_b, 2)
- Concatenate values from two fields
  - ... | eval full_name = first_name+" "+last_name
- Separate multiple eval operations with a comma
  - ... | eval full_name = last_name+", "+first_name, low_name = lower(full_name)

Eval command examples



```
index=* | eval errorType=case(status="401","Unauthorized",status="403","Forbidden",1=1,"OK")
```

Values	Count	%
OK	12,469,820	99.966%
Forbidden	4,019	0.032%
Unauthorized	259	0.002%

BOTS: https://live.splunk.com/splunk-security-dataset-project

THE UNIVERSITY OF MELBOURNE

Replace and rename commands

- Syntax: replace (<wc-string> WITH <wc-string>)... [IN <field-list>]
  - Example: replace jan* WITH Jan sat* WITH Sat IN date_month,date_wday
- Syntax: rename <wc-field> AS <wc-field>...
  - Example: rename src_* AS source_* dest_* AS destination_*

THE UNIVERSITY OF MELBOURNE

Fields command

- Adds or removes fields from search
- Syntax: fields ± <wc-field-list>
- Examples:
  - … | fields – src_port
  - "fields – src_port, dst_port" is equivalent to "fields – *_port"
- In combination with eval, fields command can be used to show internal fields
  - ... | fields + _bkt | eval bkt=_bkt

Rex command

- Rex command uses regular expressions to create new fields based on extracting patterns in other fields
- Syntax: rex [field=<field>] <regex-expression>
- The field argument is _raw by default, and specifies the field from which the new field(s) will be extracted
- regex-expression is a regular expression
- Example: extract IP address
  - … | rex field=_raw ".*(?<ip>\d+\.\d+\.\d+\.\d+)"
  - … | rex field=src_ip "\d+\.\d+\.\d+\.(?<octet>\d+)"
    | stats min(octet) as minOctet max(octet) as maxOctet
    | eval octetRange="[".minOctet.",".maxOctet."]"

A field named ip is created for events that have this pattern in their raw data

A field named octet is created for events that have the src_ip field

The new minOctet and maxOctet fields calculated using stats command can be used to find the range of the last octet in the observed IP address

Dot is used to join the results as string:

minOctet	maxOctet	octetRange
1	253	[1,253]

Regex command (filtering command)

- Regex command uses regular expressions to filter search results (it does not create new fields)
- Syntax: regex (<field>=<regex-expression> or <field>!=<regex-expression> or <regex-expression>)
  - regex “^168\.\d+\.\d+\.\d+"
  - regex src_ip!="^168\.\d+\.\d+\.\d+" | stats values(src_ip)
    - Practice! modify this command to filter private IP addresses!

values returns the list of observed values in the returned src_ip results

Summary

- Splunk Software
  - Understand Splunk architecture and what can be indexed
  - Familiar with Events & Fields, Default Fields, Data Type & Common Operators

- Search Processing Language (SPL)
  - Develop skills to use SPL for
    - Filtering Results
    - Sorting & Grouping Results
    - Filtering & Modifying Fields

References

1. https://www.splunk.com/
2. http://dev.splunk.com/view/dev-guide/SP-CAAAE3A
3. Exploring Splunk – Search Processing Language (SPL) Primer & Cookbook, David Carasso