Splunk-Cheat-Sheet

github.com/Ahmed-AL-Maghraby/SIEM-Cheat-Sheet/tree/main/Splunk-Cheat-Sheet

Top 100 Splunk Commands

Certainly, here is a comprehensive list of 100 famous Splunk SPL commands, divided into categories, along with explanations and examples for each category:

Table of contents

Table of contents

Table of contents

splunk Cheat Sheet

Basic Commands

Command	Description	Example
search	Initiates a search for events based on specified criteria	`index=web_logs status=200`

Command	Description	Example
index	Specifies the index to search within	`index=web_logs`
sourcetype	Filters events based on the specified sourcetype	`sourcetype=apache_access`

Filtering and Extraction

Command	Description	Example
where	Filters events based on conditions	`index=logs | where status="error"`
eval	Creates new fields or modifies existing ones	`index=logs | eval latency_ms=response_time/1000 | table latency_ms`
rex	Performs regular expression extraction on fields	`index=logs | rex field=message "Error: (?<error_message>.*)"`
erex	Enhanced regular expression extraction with named capture groups	`index=logs | erex "Error: (?<error_message>.*)"`

Aggregation and Statistics

Command	Description	Description
stats	Generates statistics and calculations on fields	`index=sales | stats sum(price) as total_sales by product`
timechart	Creates time-based charts and aggregates data over time	`index=web_logs | timechart count by status`
chart	Generates charts and graphs based on specified fields	`index=web_logs | chart avg(response_time) by uri`
eventstats	Performs statistics calculations on events and adds results as new fields	`index=transactions | eventstats avg(amount) as avg_amount by user`

Grouping and Transactional Analysis

Command	Description	Description
transaction	Groups related events into transactions based on conditions	`index=transactions | transaction user startswith="login" endswith="logout"`
stats count by	Counts occurrences of unique values in a field	`index=web_logs | stats count by status`
stats earliest, latest by	Retrieves the earliest and latest events for each value in a field	`index=logs | stats earliest(_time) as first_event latest(_time) as last_event by user`

Field Manipulation

Command	Description	Description
fields	Specifies fields to be included in the search results	`index=logs | fields timestamp, source, message`
rename	Renames fields in the search results	`index=logs | rename old_field as new_field`
fieldformat	Applies formatting to field values in search results	`index=metrics | eval formatted_latency = fieldformat(response_time, "duration")`
addcoltotals	Adds row and column totals to tabular search results	`index=sales | addcoltotals useother=f sum(price) as total_price`

Data Transformation

Command	Description	Description
rex mode=sed	Applies sed-like replacements using regular expressions	`index=logs | rex mode=sed field=description "s/error/warning/g"`
spath	Extracts structured data from fields containing JSON or XML	`index=logs | spath input=raw output=uri path=uri`

Command	Description	Description
spath output path	Extracts specific paths from structured data as separate fields	`index=logs | spath input=raw output=page path=uri`
spath input path output path default	Extracts structured data with default values if path is not found	`index=logs | spath input=raw output=page path=uri default="Unknown"`

Lookup and Enrichment

Command	Description	Description
lookup	Enhances data with additional information from lookup tables	`index=logs | lookup user_info.csv username as user`
inputlookup	Loads lookup data into a search	`| inputlookup user_info.csv`
outputlookup	Saves search results into a lookup file	`index=logs | stats count by user | outputlookup user_counts.csv`

Advanced Analysis

Command	Description	Description
eval case()	Performs conditional evaluation	`index=logs | eval priority = case(severity=="High", "Urgent", severity=="Medium", "Normal", true(), "Low")`
eval coalesce()	Returns the first non-null value among arguments	`index=logs | eval important_info = coalesce(critical_message, warning_message, info_message)`
eval round()	Rounds a numeric field to a specified number of decimal places	`index=metrics | eval rounded_value = round(value, 2)`
eval mvjoin()	Joins multivalue fields into a single value using a separator	`index=events | eval combined_tags = mvjoin(tags, ", ")`

Command	Description	Description
eval strftime()	Converts a Unix timestamp to a human-readable date and time format	`index=logs | eval formatted_time = strftime(_time, "%Y-%m-%d %H:%M:%S")`

Subsearch and Correlation

Command	Description	Description
subsearch	Embeds a subsearch within the main search to correlate events	`index=access_logs [ search index=error_logs | stats count ]`
tstats	Accelerated statistics command for summarizing indexed data	`| tstats count where index=web_logs by sourcetype`

Visualization and Reporting

Command	Description	Description
timechart span	Creates time-based charts with specified time spans	`index=web_logs | timechart span=1h sum(response_time)`
geostats	Generates geospatial statistics and visualizations	`index=locations | geostats count by city`
chart usenull	Includes NULL values in chart visualizations	`index=logs | chart count by user usenull=f`
rangemap	Maps field values to ranges for reporting	`index=sales | rangemap price output_field=price_range`
xyseries	Generates XY chart visualizations from multivalue fields	`index=metrics | xyseries x=time y=values`

Alerting and Monitoring

Command	Description	Description
alert	Sets up alerts based on specified conditions	`index=errors | stats count as error_count | alert threshold=100 "High Error Count"`
collect	Aggregates and stores events for future analysis	`index=access_logs | collect index=access_history`
track_alert	Tracks alert activity and results	`index=_audit action="alert_fired" | stats count by alert`

Batch Mode and Lookup

Command	Description	Description
multisearch	Runs multiple searches in parallel	`| multisearch [ search index=logs ] [ search index=metrics ]`
multisearch SID	Searches in parallel with session ID	`| multisearch SID=search1 [ search index=logs ] [ search index=metrics ]`
inputcsv	Loads data from a CSV file into the search	`| inputcsv data.csv`
inputlookup append=t	Appends data from a lookup table to the search results	`index=logs | inputlookup append=t lookup_table.csv`

Working with Time

Command	Description	Description
strptime	Converts a string to a timestamp format	`index=logs | eval event_time = strptime(timestamp, "%Y-%m-%d %H:%M:%S")`
earliest latest	Specifies time ranges for the search	`index=logs earliest=-7d latest=now`
bucket	Groups events into time buckets	`index=logs | bucket span=1h _time`

String Functions

Command	Description	Description
substr	Extracts a substring from a field's value	`index=logs | eval short_message = substr(message, 1, 50)`
len	Returns the length of a string field	`index=logs | eval message_length = len(message)`
toupper tolower	Converts string values to uppercase or lowercase	`index=logs | eval uppercase_message = toupper(message)`

Math Functions

Command	Description	Description
round	Rounds numeric values to the nearest whole number	`index=metrics | eval rounded_value = round(value)`
abs	Returns the absolute value of a number	`index=metrics | eval absolute_value = abs(change)`
sqrt	Calculates the square root of a number	`index=metrics | eval square_root = sqrt(number)`
power	Raises a number to a specified power	`index=metrics | eval squared_value = power(value, 2)`
log log10	Computes the natural logarithm or base-10 logarithm	`index=metrics | eval ln_value = log(value)`

Conditional Functions

Command	Description	Description
if()	Returns different values based on a condition	`index=logs | eval status_type = if(status>=400, "Error", "Success")`
case()	Evaluates a series of conditions and returns values accordingly	`index=logs | eval severity_level = case(severity=="High", 3, severity=="Medium", 2, severity=="Low", 1)`

Command	Description	Description
coalesce()	Returns the first non-null value among arguments	``index=logs \n

Logical Functions

Command	Description	Description
and or not	Performs logical AND, OR, and NOT operations	`index=logs | eval is_error = (severity=="High" OR status>=500)`
eval like	Matches field values with wildcard patterns	`index=logs | eval is_error = like(message, "*error*")`
mvfilter	Filters multivalue fields based on conditions	`index=events | eval tags = mvfilter(tag, like(tag, "*critical*"))`

Working with Multivalue Fields

Command	Description	Description
mvexpand	Expands multivalue fields into separate events	`index=events | mvexpand tags`
mvzip mvappend mvcombine	Manipulates multivalue fields	`index=events | eval combined_fields = mvzip(field1, field2, ", ")`
mvcount	Counts the number of values in a multivalue field	`index=events | eval tag_count = mvcount(tags)`
mvfind	Searches for values in a multivalue field	`index=events | eval has_error = mvfind(tags, "error")`

Numeric Functions

Command	Description	Description
isnull isnotnull	Checks if a field value is null or not null	`index=metrics | eval missing_value = isnull(response_time)`

Command	Description	Description
isnum	Checks if a field value is a number	`index=metrics | eval is_number = isnum(value)`
isbool	Checks if a field value is a boolean	`index=events | eval is_boolean = isbool(flag)`
mvjoin	Joins multivalue fields into a single value using a separator	`index=events | eval combined_tags = mvjoin(tags, ", ")`

Time and Date Functions

Command	Description	Description
now	Returns the current date and time	`index=logs | eval current_time = now()`
strptime strftime	Converts between Unix timestamps and human-readable dates	`index=logs | eval formatted_time = strftime(_time, "%Y-%m-%d %H:%M:%S")`
relative_time	Calculates a relative time based on a unit and offset	`index=logs earliest=relative_time(now(), "-1d@d")`
date_month date_wday	Extracts month or day of the week from timestamps	`index=logs | eval month = date_month(_time)`
now offset	Returns the current time with an offset	`index=logs | eval future_time = now() + 3600`
time	Converts a string representation of time to a Unix timestamp	`index=logs | eval event_time = time("2023-01-15 10:30:00")`
date_part	Extracts specific components (year, month, day, etc.) from a timestamp	`index=logs | eval year = date_part(_time, "year")`

IP and Geolocation Functions

Command	Description	Description
iplocation	Retrieves geolocation information for IP addresses	`index=logs | iplocation clientip`
cidrmatch	Matches IP addresses against CIDR ranges	`index=network_traffic | cidrmatch(ip, "192.168.0.0/24")`
isipv4 isipv6	Checks if a field value is an IPv4 or IPv6 address	`index=logs | eval is_ipv4 = isipv4(ip_address)`
maxmindisplocation	Retrieves geolocation information from MaxMind databases	`index=logs | maxmindisplocation ipfield=client_ip`
iptoname	Maps IP addresses to domain names	`index=network_traffic | eval hostname = iptoname(destination_ip)`

Geospatial Functions

Command	Description	Description
geostats	Generates geospatial statistics and visualizations	`index=locations | geostats count by city`
geodistance	Calculates the distance between two sets of geographic coordinates	`index=locations | eval distance_km = geodistance(lat1, lon1, lat2, lon2, "km")`
geobounds	Calculates the bounding box of a set of geographic coordinates	`index=locations | geobounds latfield=latitude lonfield=longitude`
geopoint	Converts latitude and longitude to a geopoint field	`index=locations | eval geopoint = geopoint(latitude, longitude)`
geom distance	Calculates the distance between two geopoint fields	`index=locations | eval distance_km = geom_distance(geopoint1, geopoint2, "km")`

Advanced Transformations

Command	Description	Description
spath	Extracts structured data from fields containing JSON or XML	`index=logs | spath input=raw output=uri path=uri`
spath output path	Extracts specific paths from structured data as separate fields	`index=logs | spath input=raw output=page path=uri`
spath output default	Extracts structured data with default values if path is not found	`index=logs | spath input=raw output=page path=uri default="Unknown"`
spath input path output path default	Extracts structured data with specific paths and default values	`index=logs | spath input=raw output=status_code path=code default="N/A"`

Conditional Transformations

Command	Description	Description
case()	Performs conditional evaluations and returns values	`index=logs | eval priority = case(severity=="High", "Urgent", severity=="Medium", "Normal", true(), "Low")`
if()	Returns different values based on a condition	`index=logs | eval alert_level = if(severity=="High", "Critical", "Normal")`
eval coalesce()	Returns the first non-null value among arguments	`index=logs | eval important_info = coalesce(critical_message, warning_message, info_message)`

Timechart and Chart Functions

Command	Description	Description
timechart span	Creates time-based charts with specified time spans	`index=web_logs | timechart span=1h sum(response_time)`

Command	Description	Description
chart usenull	Includes NULL values in chart visualizations	`index=logs | chart count by user usenull=f`
chart overlay	Generates overlay charts based on fields	`index=web_logs | chart count over status by host`
chart span	Creates span charts with time and non-time fields	`index=events | chart count by user span=1d`
chart stack	Generates stacked charts based on fields	`index=web_logs | chart count stack by status`
chart bins	Creates histogram-style charts with specified bin sizes	`index=metrics | chart count bins=10 by value`

Advanced Analysis and Correlation

Command	Description	Description
stats first last	Retrieves the first and last values of fields	`index=events | stats first(_time) as first_event last(_time) as last_event by user`
eventstats	Performs statistics calculations on events and adds results as new fields	`index=transactions | eventstats avg(amount) as avg_amount by user`
rare	Identifies rare values in a field	`index=errors | rare error_code`
dedup	Removes duplicate events based on specified fields	`index=logs | dedup user, ip_address`
multikv	Extracts key-value pairs from fields	`index=logs | multikv fields key1, key2`

Please note that these examples are simplified for demonstration purposes. Actual use cases might require more complex combinations of commands, functions, and field names. Replace `index`, `sourcetype`, field names, and values with your actual data and requirements. Splunk's real power comes from creatively combining these commands and functions to analyze and visualize data according to your specific use case.