Esta guía describe conceptos, características clave, comandos, y funciones de uso común de Splunk Cloud y Splunk Enterprise.

Conceptos

Eventos

Un **evento** es un conjunto de valores asociados con un registro de tiempo (timestamp). Es una sola entrada de datos y puede tener una o varias líneas. Un evento puede ser un documento de texto, un archivo de configuración, un seguimiento de pila completo, etc. Este es un ejemplo de un evento en un registro de actividad web:

```
173.26.34.223 - - [01/
Mar/2021:12:05:27 -0700] "GET
/trade/ app?action=logout HTTP/1.1"
200 2953
```

También se pueden definir transacciones para buscar y agrupar eventos que están conceptualmente relacionados durante un período de tiempo concreto. Las transacciones pueden representar actividades comerciales de varios pasos. Por ejemplo: eventos relacionados con una sola sesión de cliente en un sitio web.

Datos Métricos

Una cadena o conjunto de datos **métricos** consta de un registro de tiempo y una o más medidas. También puede contener dimensiones. Una medida es el nombre de un dato métrico y su valor numérico correspondiente. Las dimensiones proporcionan información adicional sobre las medidas. Ejemplo de cadena de datos métricos:

```
Timestamp: 08-05-2020 16:26:42.025
-0700
Measurement: metric_name:os.cpu.
user=42.12,
metric_name:max.size. kb=345
Dimensions: hq=us-west-1,
group=queue, name=azd
```

Las cadenas y eventos de datos métricos se pueden buscar y correlacionar juntos, pero se almacenan en índices de diferentes tipos, por separado.

Host, Fuente y Tipo de Fuente

Un **host** es el nombre del dispositivo físico o virtual desde donde se genera un evento. Se puede utilizar para encontrar todos los datos que se han generado en un dispositivo específico. Una fuente es el nombre del archivo, directorio, secuencia de datos u otra entrada a partir de la cual se genera un evento en particular. Las fuentes se clasifican por tipos de fuentes, que pueden tener formatos bien conocidos o formatos definidos por el usuario. Algunos tipos de fuentes comunes son los registros del servidor web HTTP y los registros de eventos de Windows.

Eventos con un mismo tipo de fuente pueden proceder de diferentes fuentes. Por ejemplo, el tipo de fuente de los eventos del archivo `source=/var/log/messages` así como los eventos de syslog con fuente `source=UDP:514`, suelen ser clasificados como `sourcetype=linux_syslog`.

Campos

Un **campo** es un conjunto de claves y valores que distinguen un evento de otro. No todos los eventos tienen las mismas claves y valores. Usando campos, puede escribir búsquedas personalizadas para encontrar eventos específicos. Cuando Splunk procesa eventos en la fase de "indexing" o durante una búsqueda, el software extrae campos en función de las definiciones en los archivos de configuración y patrones definidos por el usuario.

Utilice la herramienta "Field Extractor" para generar y validar automáticamente nuevos campos a extraer en el momento de la búsqueda utilizando expresiones regulares o delimitadores como espacios, comas, u otros caracteres.

Etiquetas

Las **etiquetas** permiten buscar eventos que contienen valores específicos. Se puede asignar una o más etiquetas a cualquier combinación de campo/valor, incluidos tipos de eventos, hosts, fuentes, y tipos de fuentes. Utilice etiquetas para agrupar valores relacionados o para realizar un seguimiento de valores abstractos, como direcciones IP o números de ID, dándoles nombres más descriptivos.

Index-Time y Search-Time

Durante la fase de "indexing", Splunk procesa los datos generados por una fuente en un host y los clasifica por tipo de fuente. También se extraen registros de tiempo (timestamps) y se separan en eventos individuales. Reglas de salto de línea se utilizan para segmentar los eventos que se mostrarán en los resultados de búsqueda. Los eventos se escriben en un índice en el disco, y pueden ser recuperados mediante búsquedas de SPL. Este proceso es conocido como **"index-time processing."**

Cuando comienza una **búsqueda,** los eventos procesados en la fase de "indexing" se recuperan del disco. Los campos del evento se extraen del texto bruto (sin formato) automáticamente. Este proceso es conocido como **"search-time processing."**

Índices

Cuando se añaden datos a la plataforma, el software de Splunk los analiza y separa en eventos individuales, extrae el registro de tiempo (timestamp), aplica reglas de salto de línea, y almacena los eventos en un índice. Nuevos índices pueden ser creados para diferentes tipos de datos, etc. Los datos se almacenan en el **índice "main" por defecto**. Los eventos se recuperan de uno o más índices mediante búsquedas de SPL.

Características Principales

Búsquedas

El método principal para encontrar y analizar eventos en Splunk es la búsqueda. Los usuarios pueden escribir una búsqueda para ver los eventos de un índice concreto, usar comandos de estadística para calcular medidas y datos métricos, generar informes, encontrar eventos con condiciones específicas en un periodo de tiempo determinado, identificar patrones en los datos, analizar tendencias y predecir cómo estas pueden cambiar en el futuro, etc. Los eventos son transformados utilizando el lenguaje de procesamiento de búsqueda de Splunk (SPL™). Las búsquedas se pueden guardar como informes o alertas, y se pueden usar en paneles de visualización.

Informes

Los **informes** son búsquedas guardadas. Los informes se pueden ejecutar ad-hoc o programar y ser ejecutados en intervalos regulares. Además, se puede configurar un informe programado para que genere alertas cuando los resultados cumplan una o más condiciones concretas. Los informes se pueden añadir a "dashboards" como paneles de visualización.

Dashboards

Los **dashboards,** o pantallas de visualización, están compuestos de paneles que pueden contener elementos como cuadros de búsqueda, campos, filtros, y visualizaciones. Los paneles de un *dashboard* suelen estar conectados a búsquedas guardadas, y pueden mostrar los resultados de búsquedas ya completadas, así como los resultados de búsquedas en tiempo real.

Alertas

Las **alertas** se activan cuando los resultados de una búsqueda cumplen una serie de condiciones específicas. Se pueden usar en búsquedas históricas o en tiempo real. Las alertas pueden ser configuradas para ejecutar ciertas acciones, como enviar información sobre la alerta por correo electrónico o mensaje instantáneo, enviar la información a través de una solicitud de HTTP, ejecutar un *script,* etc.

Características Adicionales

Conjuntos de Datos (Datasets)

Splunk permite crear y administrar diferentes tipos de **conjuntos de datos**. Algunos ejemplos incluyen archivos de búsqueda (lookups), modelos abstractos, y tablas. Las tablas son colecciones de datos seleccionados y agrupados por el usuario, y suelen ser diseñadas con un propósito específico en mente. También es posible definir y mantener grandes conjuntos de datos en tablas mediante el uso de "Table Views," una herramienta que convierte comandos de búsqueda sofisticados en interacciones simples del editor web. Esta herramienta es muy fácil de usar, incluso teniendo un conocimiento mínimo de Splunk SPL.

Modelos de Datos (Data Models)

Un **modelo de datos** es una colección de datos organizados jerárquicamente. Splunk permite hacer referencias en las búsquedas a modelos de datos enteros o a conjuntos de datos concretos dentro de uno de estos modelos. Además, es posible configurar modelos de datos con aceleración. Los modelos de datos acelerados mejoran considerablemente el rendimiento de búsqueda, razón por la cual se usan muy a

menudo en paneles de visualización e informes críticos actualizados en tiempo real.

Aplicaciones

Las **aplicaciones** son colecciones de configuraciones, elementos, vistas, y *dashboards* diseñados por usuarios de Splunk. Las aplicaciones extienden la funcionalidad del entorno básico de Splunk y permiten adaptarse a las necesidades específicas de cada equipo, como administradores de sistemas Unix o Windows, especialistas en seguridad de redes, administradores de sitios web, analistas comerciales, etc. Una sola instalación de Splunk Enterprise o Splunk Cloud puede ejecutar varias aplicaciones simultáneamente.

Búsqueda Distribuida

Una **búsqueda distribuida** proporciona una forma de modificar la escala de una implementación ya que permite separar la parte de la arquitectura de Splunk que gestiona las búsquedas y presenta los resultados (*search layer*) de la parte que se encarga de procesar datos en cada índice y recuperarlos durante las búsquedas (*indexing layer*). La búsqueda distribuida se suele utilizar para facilitar la modificación de escala horizontal y así mejorar su rendimiento, controlar el acceso a los datos procesadores por el *indexer,* y gestionar datos dispersos geográficamente.

Componentes del Sistema

Forwarder

Una instalación de Splunk que envía datos a otra. Se pueden instalar y configurar en varios dispositivos o partes de un entorno para enviar los datos generados en ese dispositivo directamente a Splunk.

Indexer

Un *indexer* es el componente de la arquitectura de Splunk que procesa y clasifica los datos que son recibidos. El *indexer* transforma los datos sin procesar en eventos y almacena los eventos en un índice. También busca los datos ya procesados y almacenados en índices durante las búsquedas. Los *"search peers"* son *indexers* que realizan búsquedas del cabezal de búsqueda (*search head*).

Search Head

En un entorno de búsqueda distribuida, el cabezal de búsqueda, o *search head*, es la parte de la arquitectura de Splunk que dirige las solicitudes de búsqueda a un conjunto de *search peers* y combina los resultados antes de mostrarlos al usuario. Un cabezal de búsqueda designado, o *dedicated search head*, es una instalación de Splunk que sólo realiza búsquedas y no procesa datos según son recibidos (*data indexing*).

Search Processing Language (SPL)

En Splunk, una búsqueda es una serie de comandos y argumentos. Los comandos se encadenan con el carácter de barra vertical, "|". Esto indica que el resultado (*output*) del comando a la izquierda de la barra vertical será utilizado (*input*) en el comando a la derecha de esta.

```
search | command1 arguments1 |
command2 arguments2 | ...
```

Al principio de cada búsqueda, hay un comando de búsqueda implícito para recuperar los eventos del índice. Las búsquedas SPL pueden incluir palabras clave, frases entre comillas, expresiones lógicas, comodines, conjuntos de nombre de campo y valor, o expresiones de comparación. El operador AND está implícito entre los términos de búsqueda. Por ejemplo:

```
sourcetype=access_combined error |
top 5 uri
```

Esta búsqueda primero recupera los eventos de actividad web de todos los índices disponibles que contienen la palabra "error" y luego devuelve los 5 valores de URI más comunes entre esos eventos.

Los comandos de búsqueda se utilizan para filtrar eventos no deseados, extraer más información, calcular valores, transformar, y analizar estadísticamente los datos ya procesados por el *indexer*. Para visualizar mejor esta idea, nos podemos imaginar que los resultados de búsqueda recuperados de los índices son escritos a una tabla dinámica. Cada evento del índice es una fila en la tabla. Los valores de cada campo son columnas en la tabla. Cada comando de búsqueda redefine el contenido y formato de la tabla. Por ejemplo, los comandos de búsqueda que filtran eventos eliminarían filas, y los comandos de búsqueda que extraen nuevos campos añadirían columnas.

Modificadores de Tiempo

Es posible especificar el marco temporal de los eventos que buscamos en Splunk utilizando los modificadores *earliest* y *latest* en nuestra búsqueda. También podemos especificar un marco temporal relativo usando una secuencia de caracteres que indique la cantidad de tiempo (valor y unidad de tiempo) y, opcionalmente, una unidad de tiempo a la que redondear. El formato a seguir sería el próximo:

```
[+|-]<integer><unit>@<snap_time_
unit>
```

Por ejemplo, la búsqueda "`error earliest=-1d@d latest=h@h`" recupera los eventos que contienen la palabra "error" y que ocurrieron entre ayer (redondeando al día, es decir, a las 00:00:00 de *ayer*) y la hora más reciente de hoy, redondeando a la hora punta.

La unidad de ajuste de tiempo redondea el tiempo hacia abajo. Por ejemplo, si son las 11:59:00 y selecciona horas (@h), la hora utilizada es 11:00:00 y no 12:00:00. También puede ajustarse a días específicos de la semana usando @w0 para referirse al domingo, @w1 para el lunes, etc.

Sub-Búsquedas

Una sub-búsqueda ejecuta su propia búsqueda y devuelve los valores de los resultados al comando principal como argumentos. La sub-búsqueda se incluye siempre entre corchetes y se ejecuta primero. Por ejemplo, la siguiente búsqueda utiliza una sub-búsqueda para encontrar todos los eventos de syslog correspondientes al usuario que tuvo el último error de inicio de sesión:

```
sourcetype=syslog [ search login
error | return 1 user ]
```

Optimización de Búsquedas

La clave para una búsqueda rápida es limitar lo máximo posible la cantidad de datos que deben extraerse del disco y luego filtrar esos datos lo antes posible en la búsqueda para que las operaciones de análisis y/o transformación se realicen sobre el mínimo número de eventos.

Es también recomendable crear índices separados en los que clasificar distintos tipos de datos. Por ejemplo, podemos configurar un índice al que enviar datos web, y otro para los datos del *firewall*.

Además, es importante especificar el marco temporal para incluir sólo los eventos necesarios. Por ejemplo, si sabemos que el evento que buscamos ha ocurrido en la última hora, usar `earliest=-1h` es mucho mejor y más eficiente que usar `earliest=-1w`, o `earliest=-1d`.

Debemos ser lo más específicos que podamos al escribir nuestras búsquedas. Por ejemplo, es mejor buscar la palabra clave `fatal_error` en vez de buscar `*error*`

En paneles de visualización, es mejor utilizar búsquedas de post-procesamiento.

También se recomienda usar *"summary indexing"* y las funciones de aceleración de informes y modelos de datos para optimizar el rendimiento de búsqueda.

Capacidades de Aprendizaje Automático - ML

Las capacidades de aprendizaje automático de Splunk están incluidas en nuestro catálogo de productos e integradas en nuestras soluciones a través de ofertas como la de Splunk Machine Learning Toolkit, Streaming ML framework, o Splunk Machine Learning Environment.

SPL2

Varios productos de Splunk han empezado a usar una nueva versión de SPL, llamada SPL2. Esta nueva versión hace que el lenguaje de búsqueda sea más fácil de usar, elimina los comandos que se usan con poca frecuencia, y mejora la consistencia de la sintaxis de los comandos. Para más información, visita SPL2 Search Reference (link al documento solo disponible en Inglés).

Comandos de Búsqueda Comunes	
Comando	**Descripción**
`chart/timechart`	Devuelve los resultados en una tabla para usar en gráficas. *Timechart* siempre utiliza el registro de tiempo (timestamp) en el eje horizontal (X-Axis).
`dedup`	Elimina los resultados subsiguientes que cumplan los criterios especificados.
`eval`	Calcula una expresión. Ver: Funciones Comunes de *eval*
`fields`	Elimina campos de los resultados de búsqueda.
`head/tail`	Devuelve los primeros/últimos N resultados.
`lookup`	Añade campos y valores de una fuente externa (documento, base de datos, etc.)
`rename`	Cambia el nombre de un campo. Se pueden usar comodines "*" para indicar varios campos.
`rex`	Específica grupos de expresiones regex para extraer campos.
`search`	Filtra los resultados y muestra solo aquellos que coinciden con la búsqueda.
`sort`	Ordena los resultados de la búsqueda según los campos especificados.
`stats`	Proporciona estadísticas. Estas pueden ser agrupadas por campos. Ver: Funciones Comunes de *stats*
`mstats`	Similar a las estadísticas pero se usa con cadenas de datos métricos en lugar de eventos.
`table`	Especifica campos a mantener en el conjunto de resultados. Devuelve los datos en formato de tabla.
`top/rare`	Muestra los valores más/menos comunes de un campo.
`transaction`	Agrupa los resultados de búsqueda en transacciones.
`where`	Filtra los resultados de búsqueda usando expresiones *eval*. Se utiliza para comparar dos campos diferentes.

Funciones Comunes de *eval*	El comando eval calcula una expresión y coloca el valor resultante en un campo (e.g. "...| eval force = mass * acceleration"). A continuación se incluye una lista de algunas de las funciones más utilizadas con el comando *eval*. También es posible utilizar operadores aritméticos básicos (+ - * / %), concatenación

	de caracteres (e.g., "...| eval name = last . "," . first"), y operaciones lógicas (AND OR NOT XOR < > <= >= != = == LIKE).

Función	Descripción	Ejemplos
abs(X)	Devuelve el valor absoluto de X.	`abs(number)`
case(X,"Y",…)	Usa los argumentos X e Y proporcionados por el usuario. Los argumentos X son expresiones lógicas. Si al evaluar X, el resultado es 1 o TRUE, el comando devuelve el valor de Y correspondiente.	`case(error == 404, "Not found", error == 500,"Internal Server Error", error == 200, "OK")`
ceil(X)	Función techo del número X.	`ceil(1.9)`
cidrmatch("X",Y)	Identifica las direcciones IP que pertenecen a una subred particular.	`cidrmatch("123.132.32.0/25",ip)`
coalesce(X,…)	Devuelve el primer valor que no es nulo.	`coalesce(null(), "Returned val", null())`
cos(X)	Calcula el coseno de X.	`n=cos(0)`
exact(X)	Evalúa una expresión X usando aritmética con formato en coma/punto flotante de doble precisión.	`exact(3.14*num)`
exp(X)	Devuelve e^X	`exp(3)`
if(X,Y,Z)	Si X es evaluado como 1 o TRUE, el resultado será el segundo argumento, Y. Si X es evaluado como 0 o FALSE, el resultado será el tercer argumento, Z.	`if(error==200, "OK", "Error")`
in(field,valuelist)	Devuelve TRUE si un valor en "value-list" coincide con un valor en "campo(field)". Esta función debe usarse dentro de la función "if".	`if(in(status, "404","500","503"),"true","false")`
isbool(X)	Devuelve TRUE si X es un operador de lógica (boolean)	`isbool(field)`
isint(X)	Devuelve TRUE si X es un número entero.	`isint(field)`
isnull(X)	Devuelve TRUE si X es NULL (nulo).	`isnull(field)`
isstr()	Devuelve TRUE si X es una secuencia de caracteres (e.g una palabra)	`isstr(field)`
len(X)	Esta función devuelve la longitud (número de caracteres) de una secuencia de caracteres X.	`len(field)`
like(X,"Y")	Devuelve TRUE si y sólo si X es igual al patrón de SQLite en Y.	`like(field, "addr%")`
log(X,Y)	Devuelve el logaritmo del primer argumento X utilizando el segundo argumento Y como base. Y por defecto es 10.	`log(number,2)`
lower(X)	Devuelve la versión minúscula del carácter X.	`lower(username)`
ltrim(X,Y)	Devuelve X con los caracteres en Y recortados del lado izquierdo. Y por defecto usa espacios y tabulaciones (tabs).	`ltrim(" ZZZabcZZ ", " Z")`
match(X,Y)	Devuelve resultados si X coincide con el patrón de expresión regex Y.	`match(field, "^\d{1,3}\.\d$")`
max(X,…)	Devuelve el máximo (de los valores proporcionados)	`max(delay, mydelay)`
md5(X)	Devuelve el hash MD5 de una secuencia de caracteres X.	`md5(field)`
min(X,…)	Devuelve el mínimo (de los valores proporcionados)	`min(delay, mydelay)`
mvcount(X)	Devuelve el número de valores de X.	`mvcount(multifield)`
mvfilter(X)	Filtra un campo de valores múltiples (multi-value field) en función de la expresión lógica X.	`mvfilter(match(email, "net$"))`
mvindex(X,Y,Z)	Devuelve el conjunto de valores de un campo de valores múltiples (multi-value field) X desde la posición de inicio (basado en cero) Y hasta la posición final Z (opcional).	`mvindex(multifield, 2)`
mvjoin(X,Y)	Usa un campo de valores múltiples X y un delimitador Y, y une los valores individuales de X usando Y.	`mvjoin(address, ";")`
now()	Devuelve la hora actual en formato Unix.	`now()`
null()	Esta función no acepta argumentos y devuelve NULL.	`null()`
nullif(X,Y)	Acepta dos argumentos, los campos X e Y, y devuelve la X si los argumentos son diferentes. De lo contrario, devuelve NULL.	`nullif(fieldA, fieldB)`
random()	Devuelve un número pseudoaleatorio que va de 0 a 2147483647..	`random()`

relative_ time (X,Y)	Acepta un registro de tiempo en formato epoch X y un argumento que especifica el tiempo relativo Y. Devuelve el valor en formato epoch de Y aplicado a X.	`relative_time(now(),"-1d@d")`
replace(X,Y,Z)	Devuelve una secuencia de caracteres formada al sustituir la secuencia Z por cada suceso de la secuencia regex Y en la secuencia X.	`Devuelve la fecha con los números de mes y día cambiados, por lo que si la entrada fue 4/30/2021 el valor de retorno seria 30/4/2021: reemplaza (date, "^(\d{1,2})/(\d{1,2})/", "\2/\1/")`
round(X,Y)	Devuelve X redondeado a la cantidad de decimales especificados por Y. El valor predeterminado es redondear a un número entero.	`round(3.5)`
rtrim(X,Y)	Devuelve X con los caracteres en Y recortados del lado derecho. Si no se especifica Y, se recortan los espacios y las tabulaciones.	`rtrim(" ZZZZabcZZ ", " Z")`
split(X,"Y")	Devuelve X como un campo de valores múltiples (multi-value field), dividido usando el delimitador Y.	`split(address, ";")`
sqrt(X)	Devuelve la raíz cuadrada de X.	`sqrt(9)`
strftime(X,Y)	Convierte el valor en formato epoch X al formato especificado por Y.	`strftime(_time, "%H:%M")`
strptime(X,Y)	Acepta un registro de tiempo representado por una secuencia de caracteres X, y devuelve el valor analizado usando el formato Y.	`strptime(timeStr, "%H:%M")`
substr(X,Y,Z)	Devuelve una sección de la secuencia de caracteres X desde la posición de inicio (basado en 1) Y para Z (opcional) caracteres.	`substr("string", 1, 3)`
time()	Devuelve la hora con una resolución de microsegundos.	`time()`
tonumber(X,Y)	Convierte la secuencia de caracteres X en un número, donde Y (opcional, el valor predeterminado es 10) define la base del número a convertir.	`tonumber("0A4",16)`
tostring(X,Y)	Devuelve un valor de campo de X como una secuencia de caracteres. Si el valor de X es un número, lo formatea como una secuencia. Si X es un valor lógico (boolean), se formatea a "TRUE" o "FALSE". Si X es un número, el segundo argumento Y es opcional y puede ser "hexadecimal" (convertir X a hexadecimal), "comas" (da formato a X con comas y 2 decimales) o "duración" (convierte segundos X al formato de hora HH:MM:SS).	`Este ejemplo devuelve : foo=615 and foo2=00:10:15:` `… | eval foo=615 | eval foo2 = tostring(foo, "duration")`

typeof(X)	Devuelve una representación de cadena del tipo de campo.	`typeof(12)+typeof("string")+typeof(TRUE)+typeof(null)`
		`Este ejemplo devuelve:`
		`"NumberStringBoolInvalid"`

urldecode(X)	Devuelve la URL X decodificada.	`urldecode("http%3A%2F%2Fwww.splunk.com%2Fdownload%3Fr%3Dheader")`
validate| (X,Y,…)	Acepta como argumentos expresiones lógicas X y secuencias de caracteres Y, y devuelve la secuencia Y correspondiente a la primera expresión X que sea evaluada como 0 o FALSE. Si todas las expresiones X son evaluadas como 1 o TRUE, devuelve NULL por defecto.	`validate(isint(port), "ERROR: El puerto no es un número entero", port >= 1 AND port <= 65535, "ERROR:` `Port está fuera de rango")`

Funciones Comunes de *stats, chart & timechart*	Funciones utilizadas con los comandos chart, stats y timechart. Los nombres de los campos pueden tener comodines, por lo que avg(*delay) podría calcular el promedio de los campos delay y xdelay.
avg(X)	Devuelve la media de los valores del campo X.
count(X)	Devuelve el número de veces que el campo X aparece en los resultados. Para indicar un valor de campo específico para que coincida, se puede usar X con formato eval(field="value").
dc(X)	Devuelve el recuento de valores distintos (no repetidos) del campo X.
earliest(X)	Devuelve el valor más antiguo de X (orden cronológico).
latest(X)	Devuelve el último valor de X (orden cronológico).
max(X)	Devuelve el valor máximo del campo X. Si los valores de X no son numéricos, el máximo se encuentra por orden alfabético.
median(X)	Devuelve la mediana del campo X.
min(X)	Devuelve el valor mínimo del campo X. Si los valores de X no son numéricos, el mínimo se encuentra por orden alfabético.
mode(X)	Devuelve el valor más frecuente del campo X.
perc<X>(Y)	Devuelve el valor del percentil X del campo Y. Por ejemplo, perc5(total) devuelve el valor del percentil 5 del campo "total".
range(X)	Devuelve la diferencia entre los valores máximo y mínimo del campo X.
stdev(X)	Devuelve la desviación estándar de la muestra del campo X.
stdevp(X)	Devuelve la desviación estándar de la población del campo X.
sum(X)	Devuelve la suma de los valores del campo X.
sumsq(X)	Devuelve la suma de los valores del campo X cuadrados.
values(X)	Devuelve la lista de todos los valores distintos del campo X. El orden de los valores es alfabético.
var(X)	Devuelve la desviación de la muestra del campo X.

Ejemplos de Búsqueda

Informes

Devuelve la media de todas las medidas métricas que terminan en cpu.percent según el nombre de las métricas. Cuenta usando un lapso de 30 segundos	`| mstats avg(_value), count(_value) WHERE metric_name="*.cpu. percent" by metric_name span=30s`
Devuelve el max(delay) de cada valor de foo dividido por el valor de bar.	`… | chart max(delay) over foo by bar`
Devuelve el max(delay) de cada valor de foo.	`… | chart max(delay) over fo o`
Cuenta los eventos según el valor del campo "host"	`… | stats count by host`
Crea una tabla que muestre el recuento de eventos y un pequeño gráfico de líneas	`… | stats sparkline count by host`
Crea un gráfico del recuento de fuentes "web" por "host" usando los valores del registro de tiempo como eje horizontal	`… | timechart count by host`
Calcula el valor medio de "CPU" cada minuto para cada "host".	`… | timechart span=1m avg(CPU) by host`

Informes (continuación)

Devuelve la media para cada hora de cualquier campo único que termine en "lay" (por ejemplo, delay, xdelay, relay, etc.).	`… | stats avg(*lay) by date_hour`
Devuelve los 20 valores más comunes del campo "url".	`… | top limit=20 url`
Devuelve los valores menos comunes del campo "url".	`… | rare url`

Informes Avanzados

Calcula la duración media general y añade 'avgdur' como un nuevo campo para cada evento que tenga el campo 'duración'	`... | eventstats avg(duration) as avgdur`
Encuentra la suma acumulativa de bytes.	`... | streamstats sum(bytes) as bytes_total | timechart max(bytes_ total)`
Encuentra anomalías en el campo 'Close_ Price' durante los últimos 10 años.	`sourcetype=nasdaq earliest=-10y | anomalydetection Close_ Price`
Crea un gráfico que muestra el recuento de eventos según pasa el tiempo, y añade a cada evento un valor pronosticado.	`... | timechart count | predict count`
Calcula la media móvil de cinco eventos con el campo 'recuento' y escribe el resultado en un nuevo campo 'recuento_suavizado'.	`"... | timechart count | trendline sma5(count) as smoothed_count"`

Datos Métricos

Devuelve una lista de todos los nombres de conjuntos de datos métricos que hay en el índice "_metrics".	`| mcatalog values(metric_ name) WHERE index=_ metrics`
Devuelve ejemplos de datos métricos almacenados en el índice "_metrics".	`| mpreview index=_ metrics target_per_ timeseries=5`
Devuelve el valor medio de un conjunto de datos métricos en el índice "_metrics". Divide los resultados en lapsos de tiempo de 30 segundos.	`| mstats avg(aws.ec2. CPUUtilization) WHERE index=_metrics span=30s`

Añadir Campos

Calcula el valor de la velocidad basado en los valores de distancia y tiempo.	`… | eval velocity=distance/time`
Extrae los campos "desde" y "hasta" utilizando expresiones regex. Si un evento sin formato contiene "De: Susan Para: David", podemos usar from=Susan and to=David.	`… | rex field=_raw "From: (?<from>.*) To: (?<to>.*)"`
Guarda el número total de eventos con el campo "recuento" en un campo llamado "total_recuento".	`… | accum count as total_count`
Para cada evento en el que exista el campo 'cuenta', calcula la diferencia entre el valor de cuenta en el evento actual y su valor anterior y guarda el resultado en un nuevo campo 'cuentadiff'.	`… | delta count as countdiff`

Filtrar Campos

Mantiene solo los campos "host" e "ip" y los devuelve en ese orden.	`… | fields + host, ip`
Elimina los campos "host" e "ip" de los resultados.	`… | fields - host, ip`

Modificar Campos

Renombra el campo "_ip" como "IPAddress".	`… | rename _ip as IPAddress`

Ejemplos de Búsqueda (continuación)

Tablas de Búsqueda (Sólo Splunk Enterprise)

Usa la tabla de búsqueda "usertogroup" para ubicar el valor de "user" correspondiente a cada evento. Envía el valor de los campos "user," "output," y "group" al evento en Splunk.	`… | lookup usertogroup user output group`
Lee la tabla de búsqueda "usertogroup" y devuelve su contenido	`… | inputlookup usertogroup`
Escribe los resultados de la búsqueda en el documento o tabla de búsqueda "users.csv"	`… | outputlookup users.csv`

Campos de Valores Múltiples

Combina los múltiples valores de el campo de destinatarios en un solo valor	`… | nomv recipients`
Separa los valores del campo "destinatarios" en múltiples valores, mostrando los principales destinatarios	`… | makemv delim="," recipients | top recipients`
Crea nuevos resultados para cada valor del campo multivalor "destinatarios"	`… | mvexpand recipients`
Encuentra el número de valores de destinatario	`… | eval to_count = mvcount(recipients)`
Encuentra la primera dirección de correo electrónico en el campo del destinatario	`… | eval recipient_first = mvindex(recipient,0)`
Encuentra todos los valores de destinatario que terminan en .net o .org	`… | eval netorg_recipients = mvfilter match(recipient,"\.net$") OR match(recipient,"\.org$"))`
Encuentra el índice de la primera coincidencia de valor del destinatario "\.org$"	`… | eval orgindex = mvfind(recipient, "\.org$")`

Expresiones Regulares (Regex)

Las expresiones regulares son útiles en varias áreas: comandos de búsqueda regex y rex; funciones de eval como match() y replace(); y en la extracción de nuevos campos.

Regex	Nota	Ejemplo	Explicación
`\s`	white space	\d\s\d	digit space digit
`\S`	No es espacio en blanco	\d\S\d	dígito dígito sin espacios en blanco
`\d`	digito	\d\d\d-\d\d-\d\d\d\d	SSN(numero de seguro social)
`\D`	no digito	\D\D\D	tres no dígitos
`\w`	carácter de palabra (letra, número o _)	\w\w\w	caracteres de tres palabras
`\W`	not a word character	\W\W\W	tres caracteres que no son palabras
`[...]`	any included character	[a-z0-9#]	cualquier carácter que sea de la a a la z, del 0 al 9 o #
`[^...]`	no included character	[^xyz]	cualquier caracter excepto x, y, o z
`*`	cero o mas	\w*	cero o más caracteres de palabras entero
`+`	uno o mas	\d+	número entero
`?`	zero o uno	\d\d\d-?\d\d-?\d\d\d\d	SSN con guiones opcionales
`|`	o	\w|\d	carácter de palabra o dígito
`(?P<var>...)`	extracción nombrada	(?P<ssn>\d\d\d-\d\d-\d\d\d\d)	sacar un SSN y asignarlo al campo 'ssn'
`(?:...)`	agrupación lógica o atómica	(?:[a-zA-Z]|\d)	carácter alfabético O un dígito
`^`	inicio de línea	^\d+	la línea comienza con al menos un dígito
`$`	fin de la línea	\d+$	la línea termina con al menos un dígito
`{...}`	number of repetitions	\d{3,5}	entre 3-5 digitos
`\`	escape	\[	escape de [character

Formato Común de Fecha y Hora

Utiliza estos valores para las funciones de evaluación strftime() y strptime(), y para los datos de eventos de marca de tiempo.

Tiempo	%H	24 houras (ceros a la izquierda) (00 to 23)
	%I	12 houras (ceros a la izquierda) (01 to 12)
	%M	Minutos (00 to 59)
	%S	Segundos (00 to 61)
	%N	subsegundos con ancho (%3N = millisecs, %6N = microsecs, %9N = nanosecs)
	%p	AM or PM
	%Z	Zona de Tiempo (EST)
	%z	Desplazamiento de la zona horaria de UTC, en horas y minutos:+hhmm or -hhmm. (-0500 for EST)
	%s	Segundos desde 1/1/1970 (1308677092)
Días	%d	Día del mes (ceros a la izquierda) (01 to 31)
	%j	Día del año (001 to 366)
	%w	Numero del día de la semana(0 to 6)
	%a	Día de la semana abreviado (Don)
	%A	Díade la semana (Domingo)
Meses	%b	Nombre del mes abreviado (Ene)
	%B	Nombre del Mes (Enero)
	%m	Numero del mes (01 to 12)
Años	%y	Año sin siglo (00 to 99)
	%Y	Año (2021)
Ejemplos	%Y-%m-%d	2021-12-31
	%y-%m-%d	21-12-31
	%b %d, %Y	Ene 24, 2021
	%B %d, %Y	Enero 24, 2021
	q|%d %b '%y = %Y-%m-%d|	q|25 Feb '21 = 2021-02-25|