このガイドでは、Splunk CloudとSplunk Enterpriseの主要な概念、機能、およびよく使われるコマンドと関数をご紹介します。

概念

イベント

イベントは、特定のタイムスタンプに関連付けられた一連の値を示します。イベントは単一のデータエントリであり、一つ、もしくは複数の行から構成されます。イベントの形式には、テキストドキュメント、設定ファイル、スタックトレース全体などがあります。次の例は、Webアクティビティログに記録されたイベントです。

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

理論的に関連があるもののタイムスタンプが異なる複数のイベントをグループ化し、サーチできるようにしたのがトランザクションです。トランザクションを使えば、複数のステップで構成されるビジネス関連のアクティビティを表すことができます。たとえば、オンラインショッピングサイトで1つのユーザーセッションに関連付けられたすべてのイベントをまとめることができます。

メトリクス

メトリクスの各データポイントは、タイムスタンプと1つ以上の測定値で構成されます。ディメンションが含まれることもあります。測定値は、メトリクス名と、対応する数値で構成されます。ディメンションは、測定値に関する追加情報を示します。メトリクスのデータポイントの例を次に示します。

```
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
```

メトリクスのデータポイントとイベントは、どちらもサーチ可能で相互に関連付けることができますが、異なるタイプのインデックスに保存されます。

ホスト、ソース、ソースタイプ

ホストは、イベントの生成元である物理デバイスまたは仮想デバイスを指します。特定のデバイスで生成されたすべてのデータを検索するために使用できます。ソースは、特定のイベントの収集元であるファイル、ディレクトリ、データストリーム、その他の情報源を指します。ソースタイプは、ソースの分類を示し、一般的なフォーマット以外に、ユーザーが定義したフォーマットを使うこともできます。一般的なソースタイプには、HTTP Webサーバーログや Windowsイベントログなどがあります。

異なるソースのイベントが同じソースタイプに分類されることもあります。たとえば、ファイル「`source=/var/log/messages`」をソースとするイベントと、syslog入力ポート「`source=UDP:514`」をソースとするイベントは、しばしば同じソースタイプ「`sourcetype=linux_syslog`」に分類されます。

フィールド

フィールドは、イベント同士を区別するための、サーチ可能な名前と値の組み合わせです。イベントによって、そのイベントが持つフィールドとフィールド値は異なります。フィールドを使えば、目的のイベントを取得するための独自のサーチを作成できます。Splunkソフトウェアでは、インデックス時やサーチ時にイベントを処理する際、設定ファイルの定義やユーザー定義のパターンに基づいてフィールドが自動的に抽出されます。

フィールド抽出機能を使うと、サーチ時に、正規表現または区切り文字(スペース、カンマなど)に基づいてフィールドが自動的に抽出、検証されます。

タグ

タグは、特定のフィールド値を含むイベントをサーチできるようにするためのナレッジオブジェクトです。イベントタイプ、ホスト、ソース、ソースタイプなど、任意のフィールド/値の組み合わせに1つ以上のタグを割り当てることができます。タグを使えば、関連するフィールド値をグループ化したり、IPアドレスやID番号にわかりやすい名前を付けて、抽象的なフィールド値を追跡したりできます。

Index-TimeとSearch-Time

Index-Timeには、まずホスト上のソースからデータが読み込まれ、適切なソースタイプに分類されます。次に、タイムスタンプが抽出され、データが解析されて個々のイベントにまとめられます。改行ルールが適用され、サーチ結果を表示するためのイベントのセグメント化が行われます。各イベントがディスク上のインデックスに書き込まれ、サーチリクエストに応じて取得できるようになります。

Search-Timeにおいてサーチが開始されると、インデックスされたイベントがディスクから取得されます。その際に、イベントの生テキストからフィールドが抽出されます。

インデックス

Splunkソフトウェアにデータを追加すると、データが解析されて個々のイベントにまとめられ、タイムスタンプが抽出されて、改行ルールが適用されます。その後、これらのイベントがインデックスに保存されます。入力ごとに新しいインデックスを作成できます。データごとに個別のインデックスを作成することが可能であり、デフォルトでは、データは「メイン」インデックスに保存されます。サーチでは1つ以上のインデックスからイベントが取得されます。

コア機能

サーチ

「サーチ」は、Splunkソフトウェアでユーザーがデータを活用するときに最もよく使われる方法です。サーチを記述してインデックスからイベントを取得する、統計コマンドを使ってメトリクスを計算しレポートを生成する、ローリングタイムウィンドウ内で特定の条件で検索を行う、データのパターンを特定する、今後のトレンドを予測するなどの操作ができます。イベントに対してこれらの操作を実行するときは、Splunk Search Process Language (SPL™)を使います。サーチは、レポートとして保存したり、ダッシュボードの強化に使用したりできます。

レポート

「レポート」は、保存済みのサーチです。必要に応じてアドホックで実行することも、スケジュールを指定して定期的に実行することもできます。スケジュールレポートを設定して、結果が特定の条件に合致したときにアラートを生成することもできます。レポートは、ダッシュボードにパネルとして追加できます。

ダッシュボード

「ダッシュボード」は、一連のパネルで構成され、各パネルには、サーチボックス、フィールド、データの視覚オブジェクトなどのモジュールが含まれます。ダッシュボードパネルは通常、保存済みサーチと連携しています。実行済みのサーチの結果を表示することも、リアルタイムサーチのデータを表示することもできます。

アラート

「アラート」は、サーチ結果が特定の条件に合致したときに生成されます。ヒストリカルサーチ(過去データに対するサーチ)とリアルタイムサーチ(リアルタイムで取り込んでいるデータに対するサーチ)のどちらでもアラートを使用できます。アラートが発生したときに、指定したメールアドレスにアラート情報を送信する、Webリソースにアラート情報を投稿するなどのアクションを実行することもできます。

その他の機能

データセット

Splunkでは、さまざまな種類の「データセット」を作成および管理できます。データセットの種類には、ルックアップ、データモデル、テーブルなどがあります。テーブルデータセットでは、特定のビジネス用途に合わせて必要なイベントデータをまとめ、整理して表示できます。テーブルビューを使えば、高度なサーチコマンドをUIエディターでのシンプルな操作に変換して、強力なテーブルデータセットを定義および管理できます。このツールは、Splunk SPLに詳しくなくても簡単に使えます。

データモデル

「データモデル」は、データセットを階層化したものです。サーチでは、データモデル全体を参照することも、データモデル内の特定のデータセットのみを参照することもできます。さらに、データモデルにはデータモデル高速化を適用できるため、サーチのパフォーマンスが大幅に向上します。データモデル高速化がダッシュボードパネルや基本的なオンデマンドレポートを強化するためによく使われるのはそのためです。

App

「App」は、設定、ナレッジオブジェクト、ユーザー設計のビューやダッシュボードを組み合わせたものです。Appを使えば、Unix/Windowsシステム管理者、ネットワークセキュリティ担当者、Webサイト管理者、ビジネスアナリストといった、様々なチームが持つ固有のニーズに合わせてSplunk環境を拡張できます。Splunk EnterpriseまたはSplunk Cloud上で複数のAppを同時に実行できます。

分散サーチ

「分散サーチ」は、サーチの管理/表示層とインデックス/サーチデータ取得層を分離することにより、導入環境をスケールする機能です。水平方向の拡張性を確保することで、パフォーマンスの向上、インデックスされたデータへのアクセス制御、地理的に分散したデータの管理が可能になります。

システムコンポーネント

フォワーダー

データを他のSplunkインスタンスに転送するSplunkインスタンスです。

インデクサー

データをインデックスするSplunkインスタンスです。インデクサーは生データをイベントに変換し、イベントをインデックスに保存します。加えて、サーチリクエストに応じて、インデックスされたデータをサーチする役目も担います。サーチヘッドからの指示でサーチを実行するインデクサーを「サーチピア」と言います。

サーチヘッド

分散サーチ環境で、サーチピアにサーチリクエストを振り分け、結果を結合してユーザーに返す役割を担うSplunkインスタンスです。サーチのみを行い、インデックスを行わない場合は、通常、「専用サーチヘッド」と呼ばれます。

Search Processing Language (SPL)

Splunkサーチは、一連のコマンドと引数で構成されます。パイプ文字「|」でによって連結することで、コマンドを連続して実行することができます。パイプは、コマンドの出力を次(右側)のコマンドに渡すことを示します。

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

サーチパイプラインの先頭には、インデックスからイベントを取得することを示す「search」コマンドがあります。これは暗黙のコマンドであり、省略可能です。サーチリクエストには、キーワード、引用符で囲んだフレーズ、ブール式、ワイルドカード、フィールド名/値ペア、比較演算子を含めることができます。複数の検索語句の間は暗黙のAND演算子によってつながれます。サーチの例を次に示します。

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

このサーチでは、インデックス済みのWebアクティビティイベントの中から、「error」という語が含まれるイベントが取得され、その中で出現頻度が高い上位5件のURI値が返されます。

サーチコマンドでは、不要なイベントのフィルタリング、詳細情報の抽出、値の計算、インデックス済みデータの変換や統計分析などの操作も行えます。インデックスから取得したサーチ結果は、動的に作成されたテーブルのようなものです。各行にはインデックス済みイベントが収められ、各列にはフィールド値が収められます。サーチコマンドを使うたびに、テーブルの内容が再定義されます。たとえば、イベントをフィルタリングするサーチコマンドでは一部の行が取り除かれ、フィールドを抽出するサーチコマンドでは列が追加されます。

時間修飾子

サーチ修飾子「latest」と「earliest」を使って、イベントを取得する時間範囲をサーチ内で指定できます。時間の長さ(整数と単位)とオプションとしてスナップ(丸める)した時間単位を示す文字列を使って、時間を相対的に指定することもできます。その構文は次のとおりです。

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

「`error earliest=-1d@d latest=-h@h`」というサーチでは、前日の始め(00:00:00)にスナップした時点から、当日の最も近い正時にスナップした時点までの間に発生した、「error」という語を含むイベントを取得しています。

スナップでは、指定した単位で時間が切り捨てられます。たとえば、現在が11:59:00の場合、時間単位(@h)でスナップすると、12:00:00ではなく11:00:00になります。また、特定の曜日にスナップすることもでき、日曜日の場合は「@w0」、月曜日の場合は「@w1」などと指定します。

サブサーチ

「サブサーチ」を使うと、サブサーチ内でのサーチ実行結果を親コマンドに引数として返すことができます。角かっこ([])で囲んだ部分がサブサーチとして先に実行されます。たとえば、次のサーチでは、サブサーチを使って、直近でログインエラーが発生したユーザーのすべてのsyslogイベントを検索しています。

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

サーチの最適化

サーチを高速化するために重要なのは、まず、ディスクから取り出すデータを必要最小限に抑えることです。その後、そのデータをできるだけ早い段階でフィルタリングして、処理する必要のあるデータをできる限り減らします。

まとめてサーチすることが少ない複数のタイプのデータがある場合、個別のインデックスに分離すべきです。たとえば、Webデータを1つのインデックスにまとめ、ファイアウォールデータを別のインデックスにまとめます。

時間範囲を必要最小限にとどめるのも効果的です。たとえば、「`-1w`」ではなく「`-1h`」や、「`earliest=-1d`」と指定します。

サーチの内容は可能な限り具体的に指定するべきです。たとえば、「`*error*`」ではなく「`fatal_error`」と指定します。

ダッシュボードのポストプロセスサーチを使用します。

サマリーインデックス化、レポート高速化、データモデル高速化などの機能も活用できます。

機械学習機能

Splunkの機械学習機能は、ポートフォリオ全体に統合され、Splunk Machine Learning Toolkitを通じて各種のSplunkソリューションに組み込まれています。

SPL2

一部のSplunk製品では、SPLの新バージョンである「SPL2」が使われています。SPL2では、サーチ言語がより使いやすくなり、ほとんど使われないコマンドが廃止され、コマンド構文の一貫性が向上しています。詳しくは、SPL2サーチリファレンスをご覧ください。

よく使われるサーチコマンド

コマンド	説明
`chart/timechart`	グラフ/時系列グラフに使える表形式の出力で結果を返します。下記の「よく使われる統計関数」を参照してください。
`dedup`	指定された条件に一致する結果が複数ある場合、2番目以降を除外します(重複削除)。
`eval`	式を計算します。下記の「よく使われるeval関数」を参照してください。
`fields`	サーチ結果から一部のフィールドを除外します。
`head/tail`	最初/最後のN件の結果を返します。
`lookup`	外部ソースのフィールド値を追加します。
`rename`	フィールドの名前を変更します。ワイルドカードを使って複数のフィールドを指定することもできます。
`rex`	指定された正規表現に名前が一致するフィールドを抽出します。
`search`	サーチ式に一致する結果のみに絞り込みます。
`sort`	指定されたフィールドに基づいてサーチ結果を並べ替えます。
`stats`	統計処理を実行します。オプションで、フィールドに基づいて結果をグループ化します。 下記の「よく使われる統計関数」を参照してください。
`mstats`	statsと似ていますが、イベントの代わりにメトリクスを対象にします。
`table`	指定されたフィールドを結果セットに含めます。データは表形式のまま出力します。
`top/rare`	フィールド内で出現頻度が最も高い/低い値を返します。
`transaction`	サーチ結果をトランザクションにまとめます。
`where`	eval式を使ってサーチ結果を絞り込みます。2つの異なるフィールドを比較するときに使用します。

Splunkが提供する包括的な製品群をご覧いただくか、以下の表を参考にして、最適なデータ活用の出発点をご検討ください。また、無料トライアル版をダウンロードして、データ戦略におけるSplunkプラットフォーム導入の効果を実際に体験することもできます。

よく使われるeval関数

evalコマンドは、式を計算し、結果の値をフィールドに格納します(例：...| eval force = mass * acceleration)。次の表に、evalコマンドとともによく使われる関数を示します。このほか、基本的な算術演算子(+ - * / %)、文字列連結(例：...| eval name = last . “,” . first)、ブール演算子(AND OR NOT XOR < > <= >= != = == LIKE)も使用できます。

関数	説明	例
`abs(X)`	Xの絶対値を返します。	`abs(number)`
`case(X,"Y",...)`	引数にXとYのペアを取り、Xに指定されたブール式を評価して、その結果がTRUEだった場合は、ペアとなるYの値を返します。	`case(error == 404, "Not found", error == 500,"Internal Server Error", error == 200, "OK")`
`ceil(X)`	数値Xを切り上げます。	`ceil(1.9)`
`cidrmatch("X",Y)`	特定のサブセットに属するIPアドレスを検索します。	`cidrmatch("123.132.32.0/25",ip)`
`coalesce(X,...)`	NULLを除く最初の値を返します。	`coalesce(null(), "Returned val", null())`
`cos(X)`	Xの余弦を計算します。	`n=cos(0)`
`exact(X)`	倍精度浮動小数点演算を使って式Xを評価します。	`exact(3.14*num)`
`exp(X)`	eを底、Xを指数とする指数関数を返します。	`exp(3)`
`if(X,Y,Z)`	Xの評価がTRUEだった場合は2番目の引数Yを返し、FALSEだった場合は3番目の引数Zを返します。	`if(error==200, "OK", "Error")`
`in(field,value-list)`	「valuelist」 内のいずれかの値が「field」の値と一致した場合にTRUEを返します。in関数は常にif関数内で使用します。	`if(in(status, “404”,”500”,”503”),”true”, ”false”)`
`isbool(X)`	Xがブール値の場合にTRUEを返します。	`isbool(field)`
`isint(X)`	Xが整数の場合にTRUEを返します。	`isint(field)`
`isnull(X)`	XがNULLの場合にTRUEを返します。	`isnull(field)`
`isstr()`	Xが文字列の場合にTRUEを返します。	`isstr(field)`
`len(X)`	文字列Xの文字数を返します。	`len(field)`
`like(X,"Y")`	Xが、Yに指定されたSQLiteパターンと類似している場合にのみTRUEを返します。	`like(field, "addr%")`
`log(X,Y)`	2番目の引数Yを底とする1番目の引数Xの対数を返します。Yのデフォルト値は10です。	`log(number,2)`
`lower(X)`	Xを小文字にします。	`lower(username)`

よく使われるeval関数(続き)

関数	説明	例
ltrim(X,Y)	Yに指定された文字をXの左側からトリムします。Yのデフォルト値はスペースとタブです。	ltrim(" ZZZabcZZ ", " Z")
match(X,Y)	Xが正規表現パターンYと一致する場合にTRUEを返します。	match(field, "^\d{1,3}\.\d$")
max(X,...)	最大値を返します。	max(delay, mydelay)
md5(X)	文字列値XのMD5ハッシュを返します。	md5(field)
min(X,...)	最小値を返します。	min(delay, mydelay)
mvcount(X)	Xの値の個数を返します。	mvcount(multifield)
mvfilter(X)	ブール式Xに基づいてマルチバリューフィールド(値を複数持つフィールド)をフィルタリングします。	mvfilter(match(email, "net$"))
mvindex(X,Y,Z)	マルチバリューフィールドXの開始位置YからZ (オプション)までのサブセットを返します。開始位置は0から始まります。	mvindex(multifield, 2)
mvjoin(X,Y)	Yに指定された区切り文字を使って、マルチバリューフィールドXの個々の値を結合します。	mvjoin(address, ";")
now()	現在の時刻をUNIX時間に変換します。	now()
null()	この関数は引数を取らず、NULLを返します。	null()
nullif(X,Y)	フィールドXとYの2つの引数を取り、引数が異なる場合はXを返し、同じ場合はNULLを返します。	nullif(fieldA, fieldB)
random()	0 ～ 2147483647の範囲の擬似乱数を返します。	random()
relative_time(X,Y)	エポックタイムXに相対時間指定子Yを適用したエポックタイム値を返します。	relative_time(now(),"-1d@d")
replace(X,Y,Z)	文字列Xに正規表現文字列Yが出現するたびに文字列Zに置き換えます。	日付表記で月と日にちを入れ替えることができます。 たとえば入力が「4/30/2022」の場合、次のように指定すると「30/4/2022」が返されます：replace(date, "^(\d{1,2})/(\d{1,2})/", "\2/\1/")
round(X,Y)	Xを小数点以下Y桁で四捨五入します。デフォルトでは整数になるように四捨五入されます。	round(3.5)
rtrim(X,Y)	Yに指定された文字をXの右側からトリムします。Yのデフォルト値はスペースとタブです。	rtrim(" ZZZZabcZZ ", " Z")

よく使われるeval関数(続き)

関数	説明	例
split(X,"Y")	Xを区切り文字Yで分割してマルチバリューフィールドに変換します。	split(address, ";")
sqrt(X)	Xの平方根を返します。	sqrt(9)
strftime(X,Y)	エポックタイム値Xを、Yで指定された形式に変換します。	strftime(_time, "%H:%M")
strptime(X,Y)	文字列Xで表された時間を、Yで指定された形式に変換します。	strptime(timeStr, "%H:%M")
substr(X,Y,Z)	部分文字列フィールドXの開始位置YからZ(オプション)までを返します。開始位置は1から始まります。	substr("string", 1, 3)
time()	実時間(ウォールクロックタイム)をマイクロ秒の分解能で返します。	time()
tonumber(X,Y)	入力文字列Xを数値に変換します。Yはオプションで、変換後の数値の基数を示し、デフォルト値は10です。	tonumber("0A4",16)
tostring(X,Y)	フィールド値Xを文字列に変換します。Xが数値の場合は文字列にフォーマット変換し、Xがブール値の場合は「TRUE」または「FALSE」にフォーマット変換します。Xが数値の場合、2番目の引数Y (オプション)には「hex」(16進数に変換)、「commas」(カンマと小数点以下2桁に変換)、または「duration」(Xを秒数として、「HH:MM:SS」の形式に変換)のいずれかを指定できます。	次の例では、foo=615, foo2=00:10:15が返されます： ... | eval foo=615 | eval foo2 = tostring(foo, “duration”)
typeof(X)	フィールドの型の文字列表現を返します。	... | eval n=typeof(12) + typeof("string") + typeof(1==2) + typeof(badfield)
urldecode(X)	Xで指定されたURLをデコードします。	urldecode("http%3A%2F%2Fwww.splunk.com%2Fdownload%3Fr%3Dheader")
validate|(X,Y,...)	ブール式Xと文字列Yをペアとして、Xが FALSEだった場合は、ペアとなるYを返します。すべてがTRUEの場合はデフォルトでNULLを返します。	validate(isint(port), "ERROR: Port is not an integer", port >= 1 AND port <= 65535, "ERROR: Port is out of range")

よく使われる統計関数

chart、stats、timechartコマンドとともによく使われる統計関数を以下に示します。フィールド名にはワイルドカードを使用できます。たとえば「avg(*delay)」と指定することで、delayフィールドとxdelayフィールドの平均を計算できます。

`avg(X)`	フィールドXの値の平均を返します。
`count(X)`	フィールドXの出現回数を返します。特定のフィールド値をカウントするには、Xを「eval(field="value")」の形式で指定します。
`dc(X)`	フィールドXの一意の値の数を返します。
`earliest(X)`	X内の時系列で最も古い値を返します。
`latest(X)`	X内の時系列で最も新しい値を返します。
`max(X)`	フィールドXの最大値を返します。Xの値が数値でない場合は、アルファベット順で最大値が決まります。
`median(X)`	フィールドXの中央値を返します。
`min(X)`	フィールドXの最小値を返します。Xの値が数値でない場合は、アルファベット順で最小値が決まります。
`mode(X)`	フィールドXで出現頻度が最も高い値を返します。
`perc<X>(Y)`	フィールドYのX番目のパーセンタイル値を返します。たとえば「perc5(total)」と指定すると、「total」フィールドの5パーセンタイル値が返されます。
`range(X)`	フィールドXの最大値と最小値の差を返します。
`stdev(X)`	フィールドXの標本標準偏差を返します。
`stdevp(X)`	フィールドXの母集団標準偏差を返します。
`sum(X)`	フィールドXの値の合計を返します。
`sumsq(X)`	フィールドXの値の平方和を返します。
`values(X)`	フィールドXのすべての値を、マルチバリューフィールドの1つのエントリとして返します。値はアルファベット順に並べられます。
`var(X)`	フィールドXのサンプル分散を返します。

サーチ例

結果のフィルタリング

Xを小数点以下Y桁で四捨五入します。デフォルトでは整数になるように四捨五入されます。	`round(3.5)`
Yに指定された文字をXの右側からトリムします。Yのデフォルト値はスペースとタブです。	`rtrim(" ZZZZabcZZ ", " Z")`
Xを区切り文字Yで分割してマルチバリューフィールドに変換します。	`split(address, ";")`
ブール式Xと文字列Yをペアとして、XがFALSEだった場合は、ペアとなるYを返します。すべてがTRUEの場合はデフォルトでNULLを返します。	`validate(isint(port), "ERROR: Port is not an integer", port >= 1 AND port <= 65535, "ERROR: Port is out of range")`

結果のグループ化

結果をクラスタリングし、「cluster_count」の値を基準に並べ替えて、データサイズの大きいクラスター上位20件を返します。	`... | cluster t=0.9 showcount=true | sort limit=20 -cluster_count`
「host」と「cookie」の値が同じで、互いに30秒以内に発生し、各イベントの間に5秒を超える中断がない結果を1つのトランザクションにまとめます。	`... | transaction host cookie maxspan=30s maxpause=5s`
IPアドレス(clientip)が同じで、最初の結果に「signon」が含まれ、最後の結果に「purchase」が含まれる結果を1つにまとめます。	`... | transaction clientip startswith="signon" endswith="purchase"`

結果の並べ替え

最初の20件の結果を返します。	`... | head 20`
結果セットの順序を逆にします。	`... | reverse`
結果を「ip」の値を基準に昇順で並べ替えてから、「url」の値を基準に降順で並べ替えます。	`... | sort ip, -url`
最後の20件の結果を逆順で返します。	`... | tail 20`

サーチ例(続き)

レポート

「cpu.percent」で終わるすべてのメトリクスをメトリクス名ごとにまとめ、30秒間隔で平均値と件数を集計します。	`| mstats avg(_value), count(_value) WHERE metric_name="*.cpu.percent" by metric_name span=30s`
「bar」の値でまとめた「foo」の値ごとに「delay」の最大値を表示します。	`... | chart max(delay) over foo by bar`
「foo」の値ごとに「delay」の最大値を表示します。	`... | chart max(delay) over foo`
「host」ごとにイベント数をカウントします。	`... | stats count by host`
イベント数と小さい折れ線グラフを含む表を作成します。	`... | stats sparkline count by host`
「host」ごとのカウントを示すタイムチャートを作成します。	`... | timechart count by host`
「host」ごとに1分間隔の「CPU」の平均値を示すタイムチャートを作成します。	`... | timechart span=1m avg(CPU) by host`
文字列「lay」で終わる一意のフィールド(delay、xdelay、relayなど)の1時間間隔の平均値を返します。	`... | stats avg(*lay) by date_hour`
「url」フィールドで出現頻度が高い値上位20件を返します。	`... | top limit=20 url`
「url」フィールドで出現頻度が最も低い値を返します。	`... | rare url`

高度なレポート

「duration」全体の平均を計算し、「duration」フィールドがある各イベントに「avgdur」という名前の新しいフィールドを追加して平均値を格納します。	`... | eventstats avg(duration) as avgdur`
「bytes」の累計を表示します。	`... | streamstats sum(bytes) as bytes_total | timechart max(bytes_total)`
「Close_Price」フィールドでの過去10年間における外れ値を検索します。	`sourcetype=nasdaq earliest=-10y | anomalydetection Close_Price`
イベント数と各イベントの予測値および範囲を時系列で示すグラフを作成します。	`... | timechart count | predict count`
「count」フィールドの最後5つのイベントの単純移動平均を計算し、「smoothed_count」という名前の新しいフィールドに格納します。	`"... | timechart count | trendline sma5(count) as smoothed_count"`

サーチ例(続き)

メトリクス

「_metrics」メトリクスインデックス内のすべてのメトリクス名をリストします。	`| mcatalog values(metric_name) WHERE index=_metrics`
「_metrics」メトリクスインデックスに保存されたメトリクスデータポイントの一部を表示します。	`| mpreview index=_metrics target_per_timeseries=5`
「_metrics」メトリクスインデックス内の特定のメトリクスの平均値を返します。平均は30秒間隔で計算します。	`| mstats avg(aws.ec2.CPUUtilization) WHERE index=_metrics span=30s`

フィールドの追加

距離を時間で割った結果を「velocity」に格納します。	`... | eval velocity=distance/time`
正規表現を使って「from」フィールドと「to」フィールドを抽出します。たとえば、イベントの生データに「From: Susan To: David」という文字列が含まれている場合、「from」に「Susan」、「to」に「David」が入ります。	`... | rex field=_raw "From: (?<from>.*) To: (?<to>.*)"`
「count」の累計を「total_count」という名前のフィールドに格納します。	`... | accum count as total_count`
「count」フィールドがあるイベントが見つかるたびに、その値と前のイベントの同フィールドの値の差を計算し、結果を「countdiff」に格納します。	`... | delta count as countdiff`

フィールドのフィルタリング

「host」フィールドと「ip」フィールドのみを、その順序で表示します。	`... | fields + host, ip`
「host」フィールドと「ip」フィールドを結果から除外します。	`... | fields - host, ip`

ルックアップテーブル(Splunk Enterpriseのみ)

各イベント内で、ルックアップテーブル「usertogroup」の「user」の値と一致する値を探します。見つかったら「group」フィールドの値をイベントに追加します。	`... | lookup usertogroup user output group`
「transforms.conf」ファイルで定義された「usertogroup」ルックアップテーブルを読み込みます。	`... | inputlookup usertogroup`
サーチ結果を「users.csv」ルックアップファイルに書き出します。	`... | outputlookup users.csv`

フィールドの変更

「_ip」フィールドの名前を「IPAddress」に変更します。	`... | rename _ip as IPAddress`

サーチ例(続き)

正規表現

正規表現は、サーチコマンドのregexとrex、eval関数のmatch()とreplace()、フィールド抽出など、さまざまな場面で役立ちます。

正規表現	説明	例	例の説明
\s	空白文字	\d\s\d	数字 空白文字 数字
\S	空白文字以外	\d\S\d	数字 空白文字以外 数字
\d	数字	\d\d\d-\d\d-\d\d\d\d	SSN
\D	数字以外	\D\D\D	数字以外の3文字
\w	単語文字(アルファベット、数字、 下線)	\w\w\w	単語文字3文字
\W	単語文字以外	\W\W\W	単語文字以外の3文字
[...]	かっこ内のいずれかの文字	[a-z0-9#]	a〜z、0〜9、#のいずれかの文字
[^...]	かっこ内の文字以外	[^xyz]	x、y、z以外の文字
*	0以上	\w*	単語文字0文字以上
+	1以上	\d+	整数
?	0または1	\d\d\d-?\d\d-?\d\d\d\d	ダッシュを含まない形式を含むSSN
|	または	\w|\d	単語文字または数字
(?P<var>...)	名前付きの抽出	(?P<ssn>\d\d\d-\d\d-\d\d\d\d)	SSNを抽出して「ssn」フィールドに割り当て
(?: ...)	論理(アトミック)グループ化	(?:[a-zA-Z]|\d)	アルファベット文字または数字
^	行の先頭	^\d+	数字1文字以上で始まる行
$	行の末尾	\d+$	数字1文字以上で終わる行
{...}	反復回数	\d{3,5}	3 〜 5桁の数字
\	エスケープ	\[	「[」文字をエスケープ

マルチバリューフィールド

「recipients」フィールドの複数の値を1つの値に結合します。	`... | nomv recipients`
「recipients」 フィールドの値を複数のフィールド値に分割し、出現頻度が最も高い受信者を表示します。	`... | makemv delim="," recipients | top recipients`
マルチバリューフィールド「recipients」 の値ごとに新しい結果を作成します。	`... | mvexpand recipients`
「recipients」 フィールドの値の個数を調べます。	`... | eval to_count = mvcount(recipients)`
「recipient」フィールドに含まれる最初の値を抽出します。	`... | eval recipient_first = mvindex(recipient,0)`
値が「.net」または「.org」で終わるすべての値を抽出します。	`... | eval netorg_recipients = mvfilter match(recipient,"\.net$") OR match(recipient,"\.org$"))`
マルチバリューフィールド内で最初に「\.org$」と一致する値の、フィールド内での順番(インデックス)を抽出します。	`... | eval orgindex = mvfind(recipient, "\.org$")`

サーチ例(続き)

よく使われる日付/時刻フォーマット

以下のフォーマットは、eval関数のstrftime()とstrptime()、およびイベントデータへのタイムスタンプ追加でよく使われます。

時刻	%H	2桁表記の24時間制の時間(00 ～ 23)
	%I	2桁表記の12時間制の時間(01 ～ 12)
	%M	分(00 ～ 59)
	%S	秒(00 ～ 61)
	%N	1秒未満(%3N = ミリ秒、%6N = マイクロ秒、%9N = ナノ秒)
	%p	午前または午後
	%Z	タイムゾーン(EST)
	%z	UTCからのタイムゾーン補正、時間と分の形式は「+hhmm」または「-hhmm」(ESTの場合は「-0500」)
	%s	1970年1月1日(1308677092)からの秒数
日	%d	2桁表記の日にち(01 ～ 31)
	%j	1年の通算日数(DOY) (001 ～ 366)
	%w	曜日(0 ～ 6)
	%a	曜日の略称(Sun)
	%A	曜日(Sunday)
月	%b	月名の略称(Jan)
	%B	月名(January)
	%m	数字表記の月(01 ～ 12)
年	%y	2桁表記の西暦年(00 ～ 99)
	%Y	4桁表記の西暦年(2022)
例	%Y-%m-%d	2022-12-31
	%y-%m-%d	22-12-31
	%b %d, %Y	Jan 24, 2022
	%B %d, %Y	January 24, 2022
	q|%d %b '%y = %Y-%m-%d|	q|25 Feb '22 = 2022-02-25|