

SPL2

Powering the next generation of Search

FN1740





Alex James

Sr. Principal Architect | Splunk



Andrew Peters

Sr. Principal Engineer | Splunk

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

Splunk, Splunk>, Listen to Your Data, The Engine for Machine Data, Splunk Cloud, Splunk Light and SPL are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names, or trademarks belong to their respective owners. © 2019 Splunk Inc. All rights reserved.

Agenda

What are we going to cover today?

- Why SPL2
- Goals & Principles
- Datasets over Commands
- Language Features
- SQL & Fluent SQL

SPL2 Goals & Principles



Why SPL2?

- Insight comes when Splunk answers complex questions
- However complex Questions require deep knowledge of SPL
- SPL experts become a bottleneck

- SPL2 aims to simplify SPL
- ... and reduce or remove the bottleneck

Goals of SPL2

- **Simplicity**
Smaller, more composable, more intuitive
- **Familiarity**
Both for SPL1 and SQL users
- **Consistency**
Expressions are uniform, literals are everywhere
- **Extensibility & Power**
Datasets are the ultimate hook
- **Security**
Controlled ways to elevate privileges

Key Design Principles

- Embrace familiarity
 - SPL1 → search index="main"
 - SQL → select * from main
- Smaller set of more general and more powerful commands
 - Collapse similar commands into one more powerful command
 - Example: Grouping by time can now be done directly in stats (no need for bin).
- Instead of new commands favor new datasets if possible
 - You read FROM a dataset
 - You optionally write INTO a dataset

SPL2 Commands

Think of this as a subset of classic SPL



SCP ships with a number of built-in commands that can be composed to create searches. Many less commands than classic SPL.

bin	branch	dedup	eval	fields
from	head	into	join	lookup
mvexpand	rename	reverse	rex	search
sort	stats	streamstats	thru	timechart
timewrap	union	where	apply	fit

untable	xyseries	transaction	???	
---------	----------	-------------	-----	--

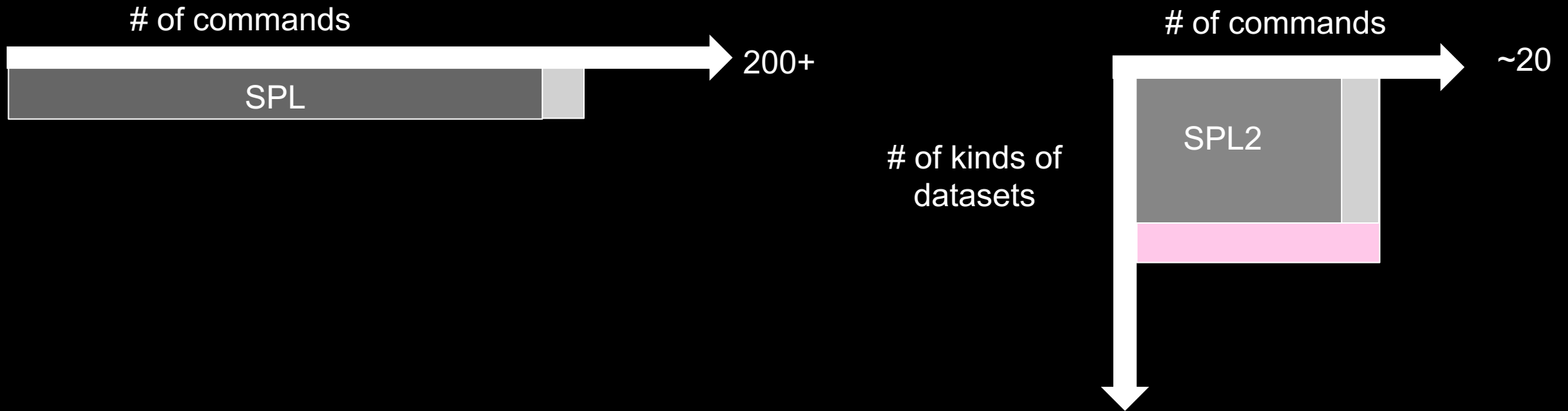
*ML API Commands

**Commands Coming Soon

Datasets over Commands



SPL1 vs SPL2



From

The Heart of SPL2

- **FROM command syntax**

from <dataset> [**as** <alias>]
[**join** <dataset2> **as** <alias2> **on** <constraint>]
[**where** <filter>]
[**group by** <fields>]
[**select** <expressions>]
[**having** <post-filter>]
[**orderby** <fields>]
[**offset** <integer>]
[**limit** <integer>]

- **Example**

from main
where action="logon"
group by host
select count(), host
order by count

→ **search**, **savedsearch**, **mstats**, **inputlookup** ...
→ **join** or **lookup** command
→ **where** or **search** command
→ **stats**
→ **fields** / **eval**
→ **where** or **search** command
→ **sort**
→ **streamstats** + **where**
→ **head**

→ **search** index="main" action="logon"
| **stats** count() by host
| **sort** -count

From Examples

- **search** index=main status=401
- | **tstats** count **WHERE** index=main **BY** host
- | **mstats** avg(aws.ec2.CPUUtilization)
WHERE index=mymetricsdata
BY host, span=1m
- | **inputlookup** knownbots **where**
countrycode=408
- | **mcatalog** values(ip)
where metric_name=login_failure
- | **loadjob** 1233886270.2
- | **makeresults** count=2
- **from** main **where** status=401
- **from** main **group by** host **select** count(), host
- **from** mymetricsdata
group by host, span(_time, 1m)
select avg(aws.ec2.CPUUtilization), host, _time
- **from** knownbots
where countrycode=408
- **from** catalog.metrics
where metric_name="login_failure"
select values(ip)
- **from** sid_1233886270_2
- **from** [{}, {}]

Intro & Lookup Examples

... | **outputlookup** mylookup

... | **collect** index=myindex

... | **meventcollect** <args>

... | **sendemail** <args>

... | **lookup** users uid as id

... | **iplocation** srcip

... | **geom** geohex

... | **into** mylookup

... | **into** myindex

for now:

... | **eval** index="myindex" | **into** ingest.events

... | **into** mymetricsindex

for now:

... | **eval** index="mymetricsindex" | **into** ingest.metrics

... | **into** emails

for now:

... | **eval** trigger="email" | **into** actions

... | **lookup** users uid as id

... | **lookup** geo.iplocation srcip AS ip

... | **lookup** geo.geom featureId

Literals & Comments



SPL2 Literals

Strings and regular expressions

- Strings

“Regular string literals now support escape sequences like \n and \t!”

- Raw strings

@“this is a raw string literal, \n will not be interpreted as a newline”

- Regular expressions (PCRE)

/s+(?<ports>port \d+)/i

- Times

from mydataset **where** earliest=-4y@q

<spl> | **stats** sparkline(count(error), 1day)

Comments

Inline documentation in SPL2

- SPL2 supports both line and block style comments

`/*`

Block comments look like this, and
can be multi-line.

`*/`

`--` Line comments are like SQL...



Demo

Parameters & JSON & Dataset Literals



Parameters

Creating re-usable SPL

- Parameters are special variables whose value is provided “externally”
from main where user = **\$current_user**
- Values MUST be provided when executing via **queryParameters** property
- Built-in Parameters:
 - **\$earliest** – Start of the time range
 - **\$latest** – End of the time range
 - **\$relativeTimeAnchor** – The base time used when computing relative time ranges
 - **\$timezone** – The timezone used when snapping to the start of the day i.e. @d

Dataset Literals

- Filtering

```
from [{host: "127.0.42.1", user: "Alex"},  
      {host: "127.0.42.2", user: "Andrew"}]  
where user="Alex"
```



```
[  
  { host: "127.0.42.1", user: "Alex" }  
]
```

- Projecting

```
from [{host: "127.0.42.1", user: "Alex"},  
      {host: "127.0.42.2", user: "Andrew"}]  
select host
```



```
[  
  { host: "127.0.42.1" }  
  { host: "127.0.42.2" }  
]
```

- Aggregating

```
from [{host: "127.0.42.1", user: "Alex"},  
      {host: "127.0.42.2", user: "Andrew"}]  
select count()
```



```
[  
  { count: 2 }  
]
```


Enhanced JSON Support

- JSON Functions

- `json_array(..)` → create a JSON Array
- `json_object(..)` → create a JSON Object (or Map)
- `json_extract(..)` → extract one or more properties from a JSON Object
- `json_set(..)` → set one or more values in an existing JSON Object
- `json_delete(..)` → delete a property from a JSON Object

- JSON Array Literals

- `<spl> | eval a=1, b=2, c= [a, a+b, 10]` → `c = [1,3, 10]`

- JSON Object Literals

- `<spl> | eval a=1, b=2, c = {x:a,y:a+b,z:10}` → `c = {"x": 1, "y": 3, "z": 10}`

- De-ref JSON Property

- `... | eval d = c.x + c.y + c.z` → `d = 14`



Demo

SQL

Providing a SQL-like experience in SPL2



SQL

- SPL2 is a superset of SQL Query
 - **from** webtraffic as w
join users as u **on** w.uid = u.id
group by u.department
select count(), u.department
→
 - **select** count(), u.department
from webtraffic as w
join users as u **on** w.uid = u.id
group by u.department
- Fluent SQL support
 - **<spl> | where** user.department="Sales" **limit** 10
 - →
 - **from** [**<spl>**] **where** user.department="Sales" **limit** 10
 - →
 - **select * from** [**<spl>**] **where** user.department="Sales" **limit** 10



Demo

Key Takeaways

SPL2 is a modern
language for
processing data at
scale

1. SPL2 is concise, simpler but no less powerful than SPL
2. SPL2 will be familiar to both SPL and SQL users
3. SPL2 fully embraces datasets



**Thank
You!**