

# Using Splunk Data Stream Processor as a Data Transformation, Alerting and Action Engine

Dirk Nitschke | Bashar Abdul-Jawad





**Dirk Nitschke**

Staff Sales Engineer | Splunk



**Bashar Abdul-Jawad**

Principal Software Engineer | Splunk



# Forward-Looking Statements



During the course of this presentation, we may make forward-looking statements regarding future events or plans 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 may differ materially. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, it may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements made herein.

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 functionalities described or to include any such feature or functionality in a future release.

Splunk, Splunk>, Turn Data Into Doing, 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

1. Introduction to Splunk Data Stream Processor
2. Data Transformation
3. Alerting and Actions



# Introduction to Splunk Data Stream Processor

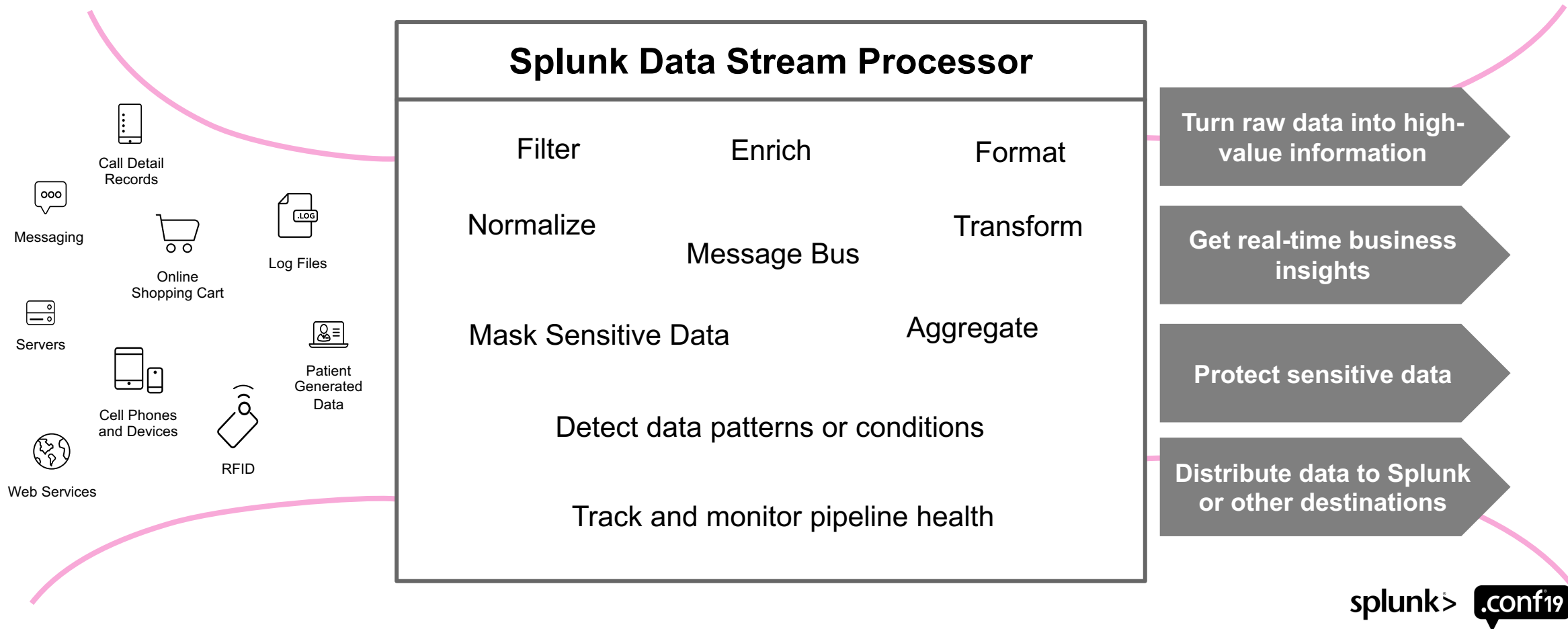
---

## Overview



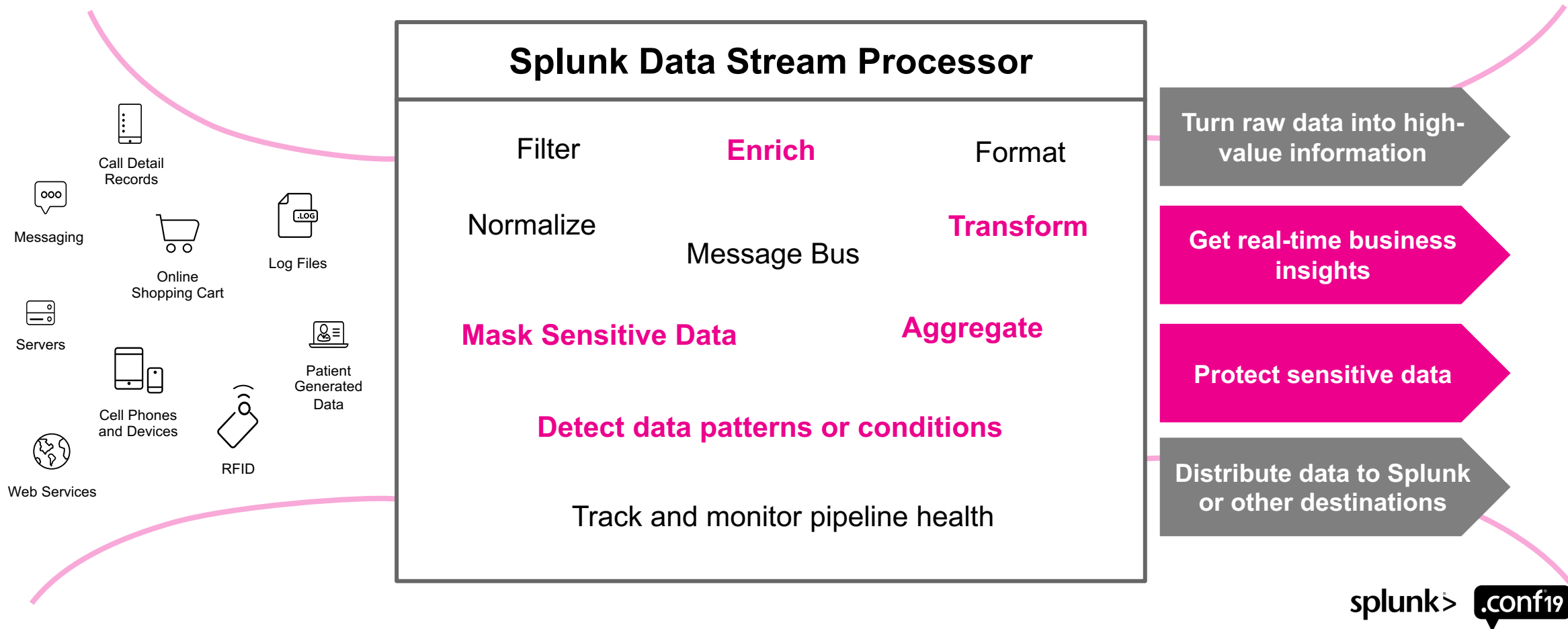
# Splunk Data Stream Processor

A real-time streaming solution that collects, processes, and delivers data to multiple destinations in milliseconds



# Splunk Data Stream Processor

A real-time streaming solution that collects, processes, and delivers data to multiple destinations in milliseconds





# Data Transformation

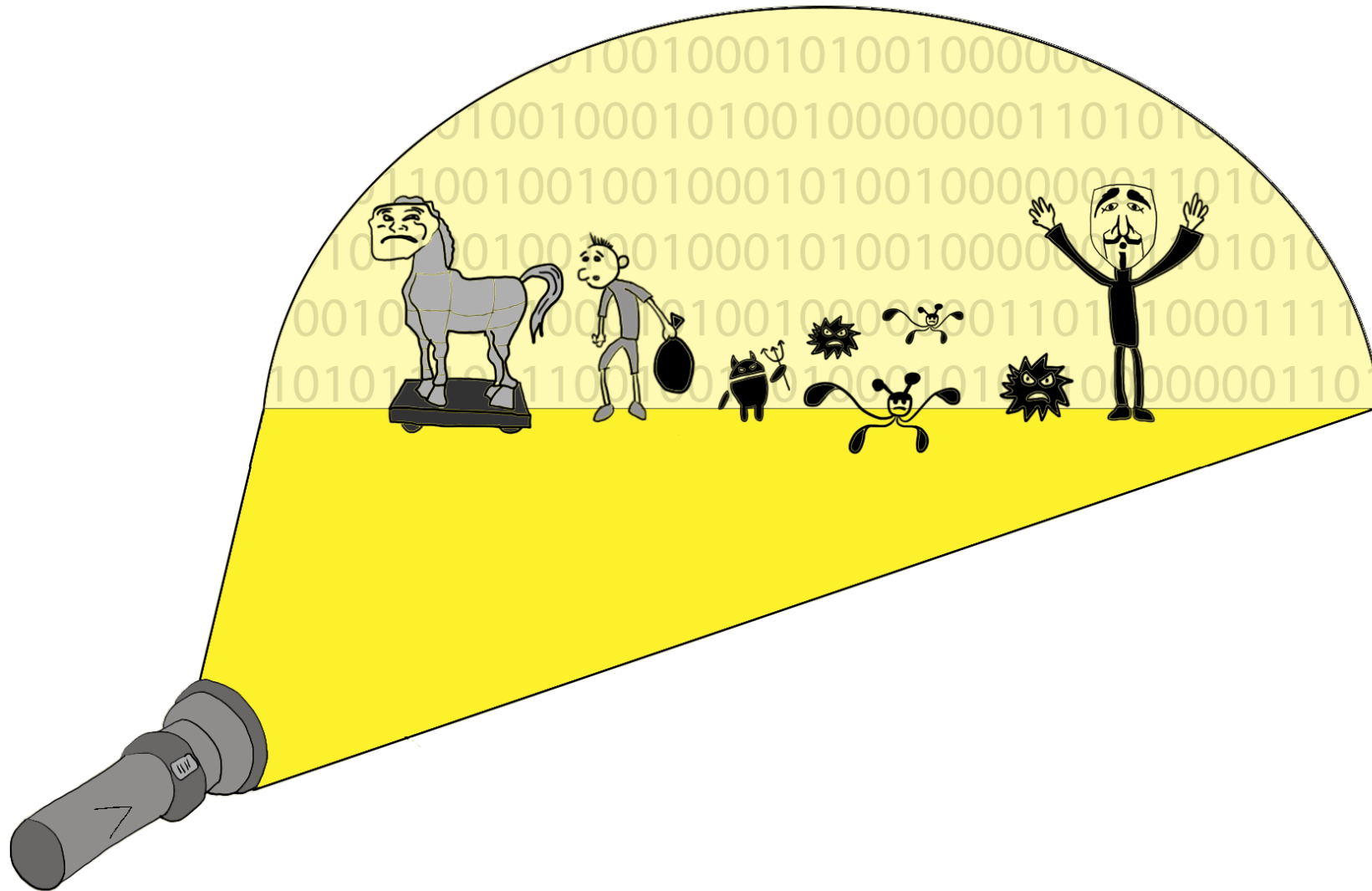
---

Do you have something to hide?

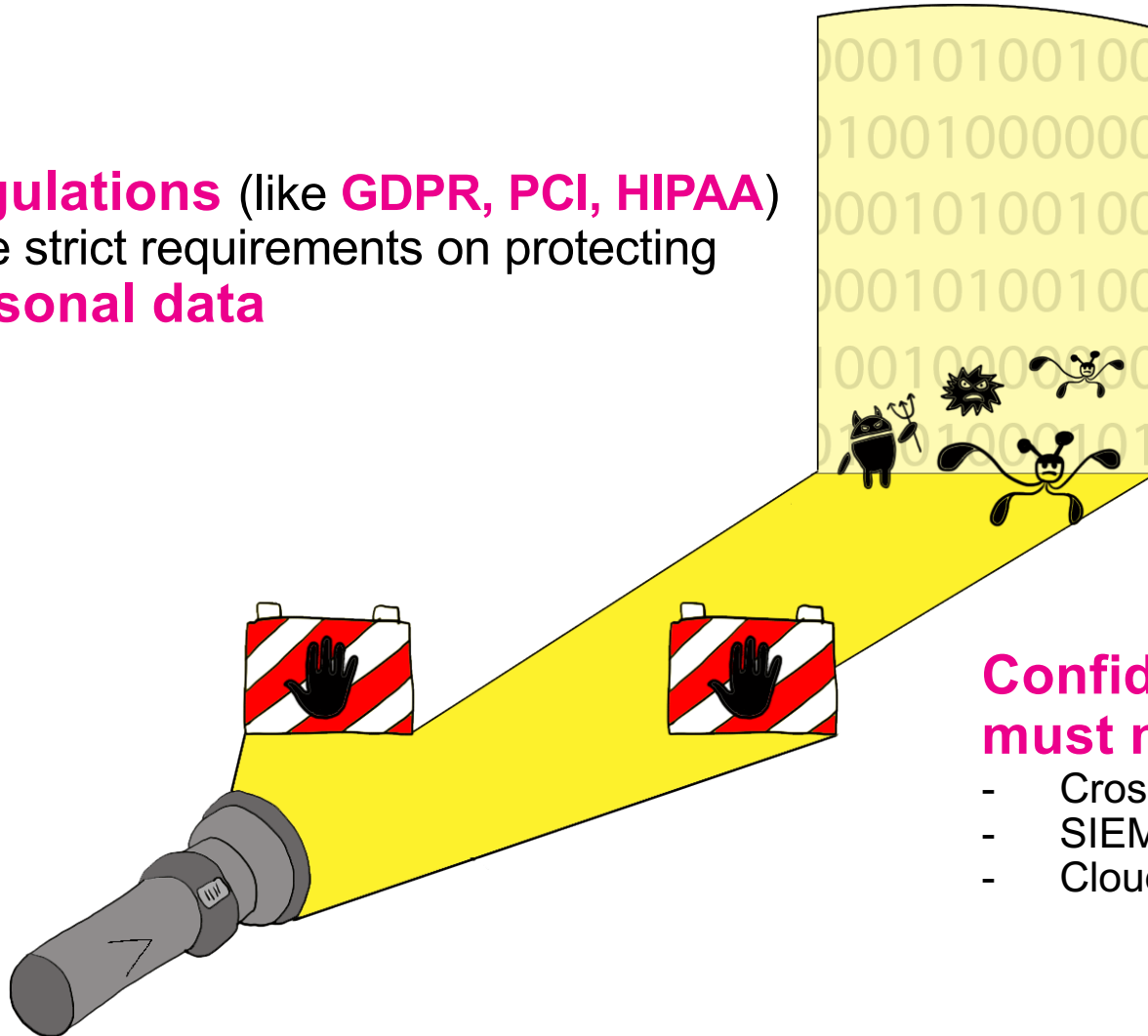


# Why You Need Data Transformation

```
2019-10-23T09:03:22+0200 src_ip=192.168.178.22 dest_ip=192.168.1.223  
user=dnitschke action=purchase cc=9877224571829008
```



**Regulations** (like **GDPR, PCI, HIPAA**)  
have strict requirements on protecting  
**personal data**



**Confidential data** in events  
**must not be shared / collected**

- Cross border organization
- SIEM as a Service
- Cloud



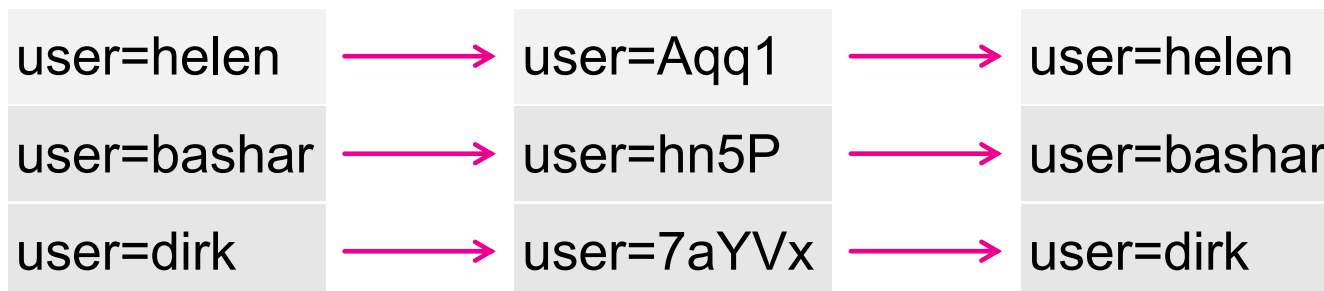
# Typical Approaches to Data Masking

## Anonymization and Pseudonymization

Anonymization: Irreversibly replace data



Pseudonymization: Reversibly replace data



# There Ain't No Such Thing as a Free Lunch

## Anonymization

- Less granularity
- Can't get back the original data

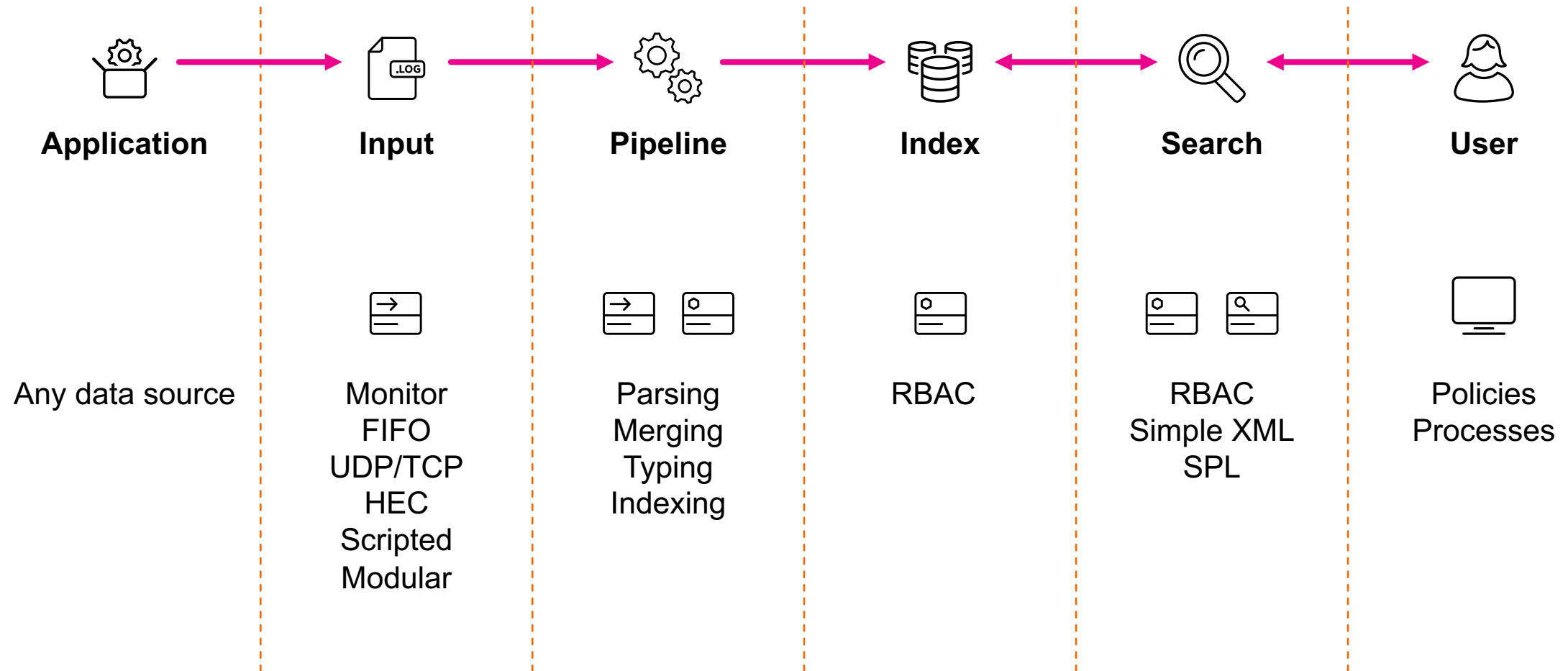
## Data Format

- May have to preserve the data format for analysis

## Keep it simple

- Find the right balance between protection and analytics requirements

# Got It – but Where and How?



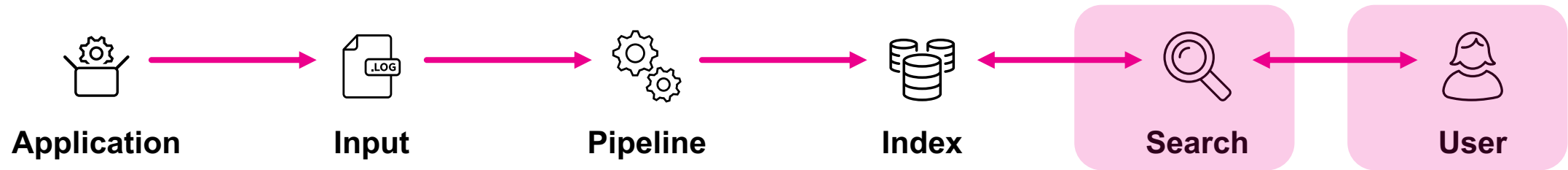
# Prevention Is Better Than Cure



- Protect data at earliest stage in the process
- Data source owner is responsible
- Application support
- May need a means to re-identify data again

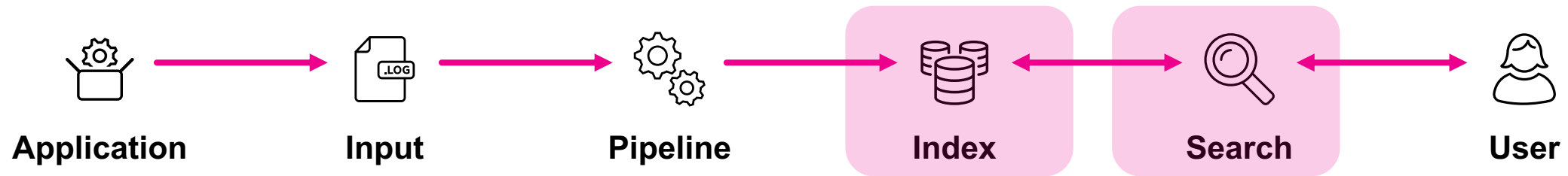


# At the 11<sup>th</sup> Hour



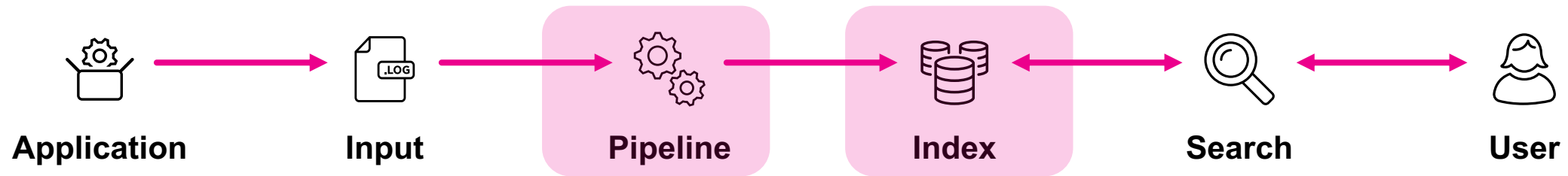
- Hide data at presentation layer
  - `<your_search> | eval user=sha256(user)`
  - Can use your own custom search command
- Optionally: user lockdown
  - Pre-defined app with dashboard access only
  - No search app, no raw search, no raw event drill down
- May be good enough

# Ingest Now, Transform Later



- Scheduled search selects and transforms the data
  - SPL and custom search commands if needed
- Send modified events to a different index
  - Think about collect, cefout, or a custom search command
- Modified events are delayed because of scheduling
- Preservation of event metadata

# At Ingestion



- Typing pipeline: regex replacement processor
- Use SEDCMD, TRANSFORMS, INGEST\_EVAL to modify data at indexing time
- Easy to implement
- SEDCMD, TRANSFORMS use regular expressions
- INGEST\_EVAL can do (almost) everything eval can do at search time

INGEST\_EVAL is available since Splunk Enterprise version 7.2

# Before Ingestion

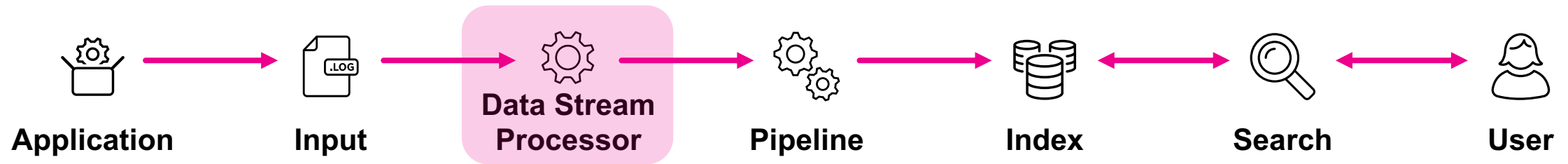


- Pre-process data
- Create your own data input capabilities using a modular input or even use `unarchive_cmd`
- Flexible but requires scripting, programming
- Strongly depends on type of input

# What If You Could...

- Benefit from existing input methods?
- Offload work from the indexing layer?
- Apply a custom function to your data?
- Enrich data at ingestion?
- Have better visibility into what's going on?

# Enter: Splunk Data Stream Processor

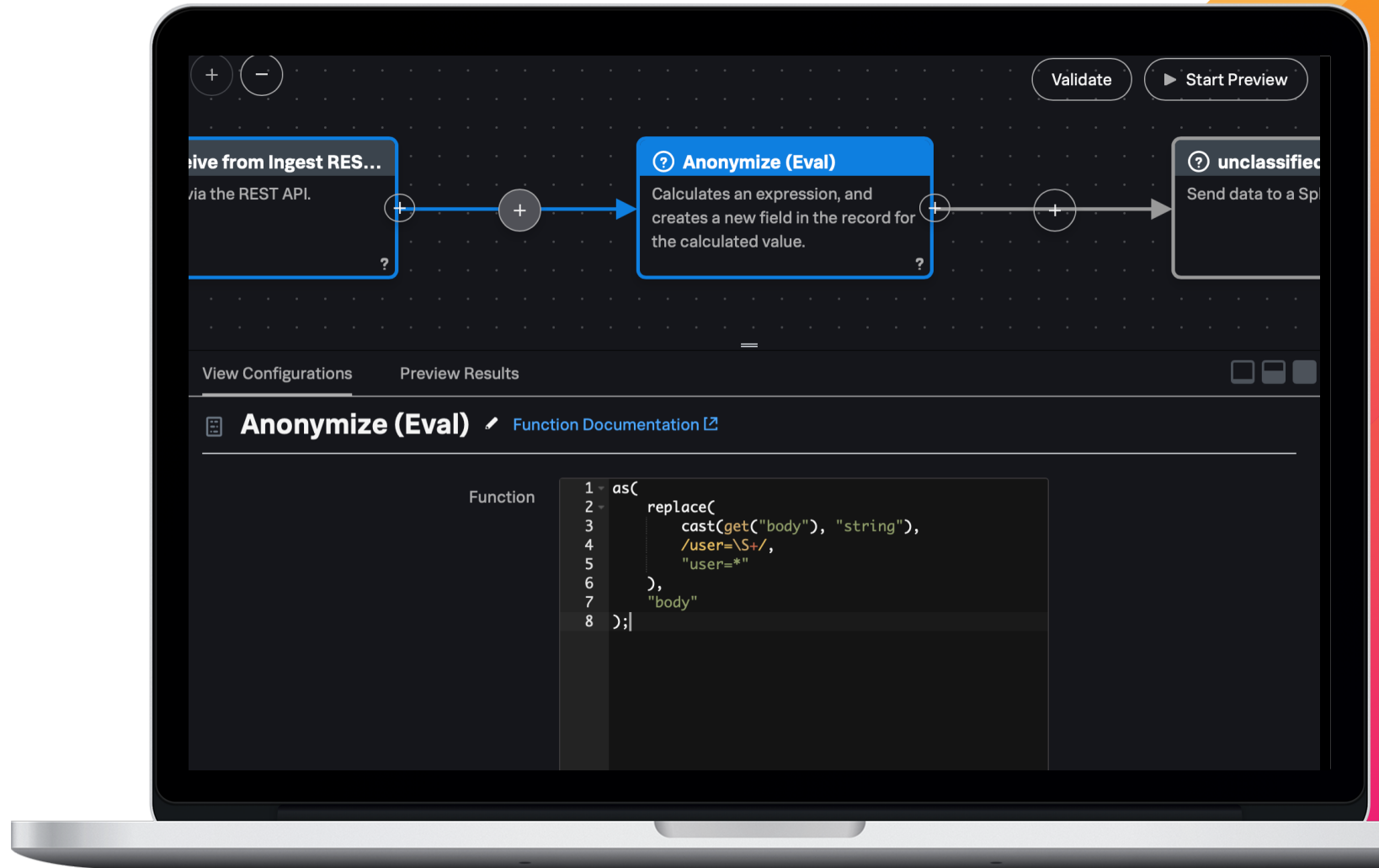


- Supports different inputs and outputs
- Graphical user interface, preview steps in pipeline
- Data transformation, enrichment, routing
- Use your own functions

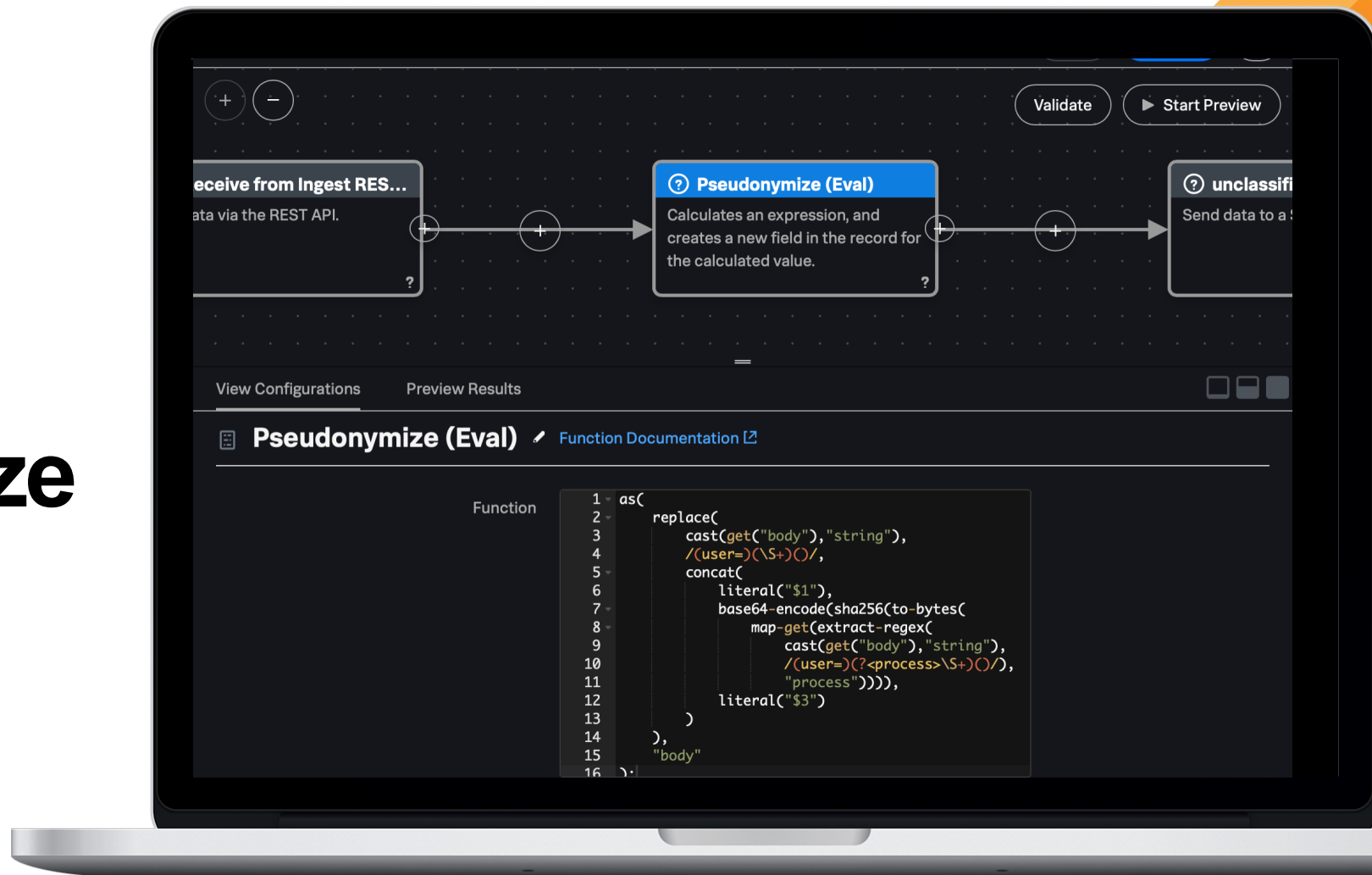


# Demo

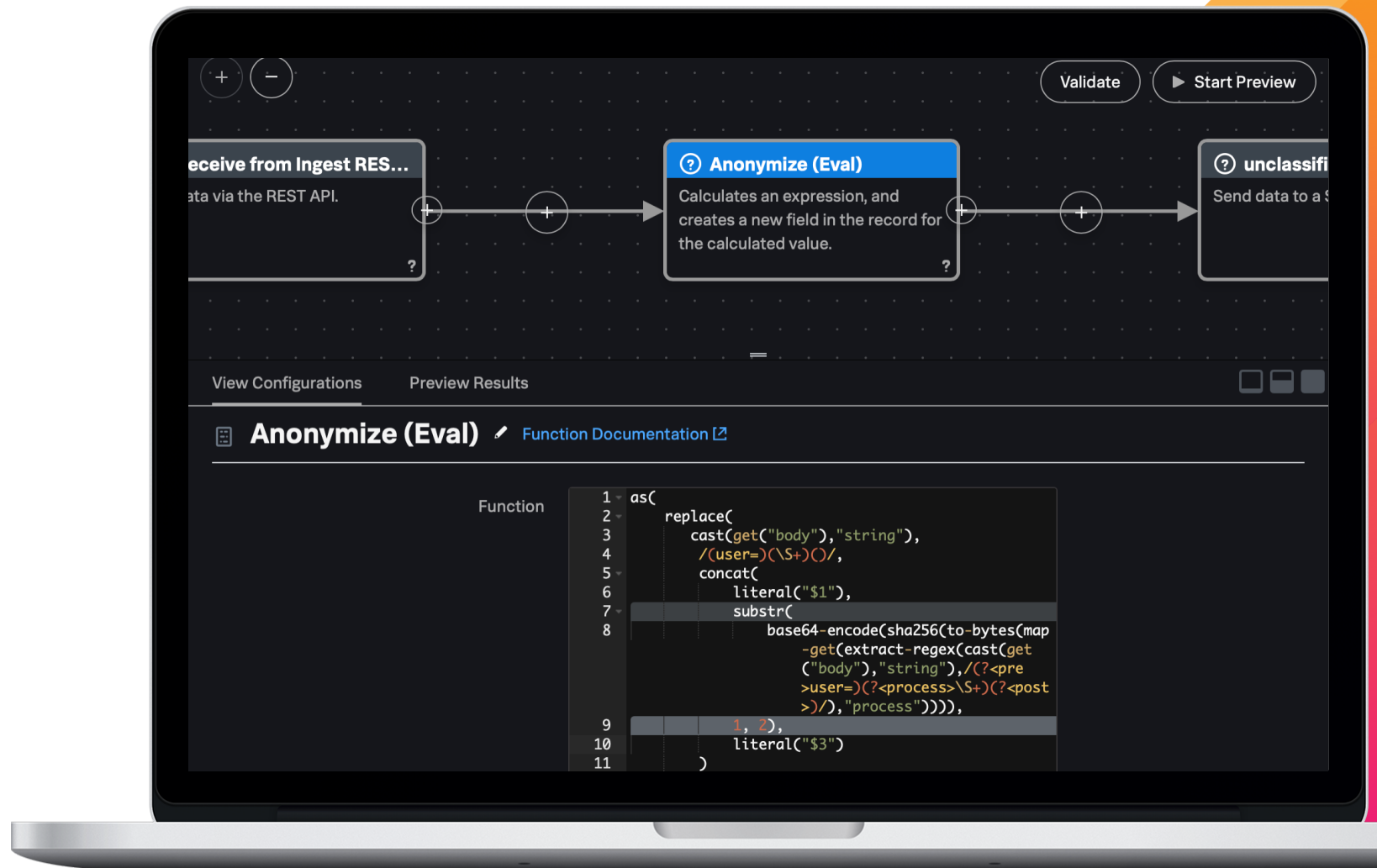
# Anonymize (Single Category)



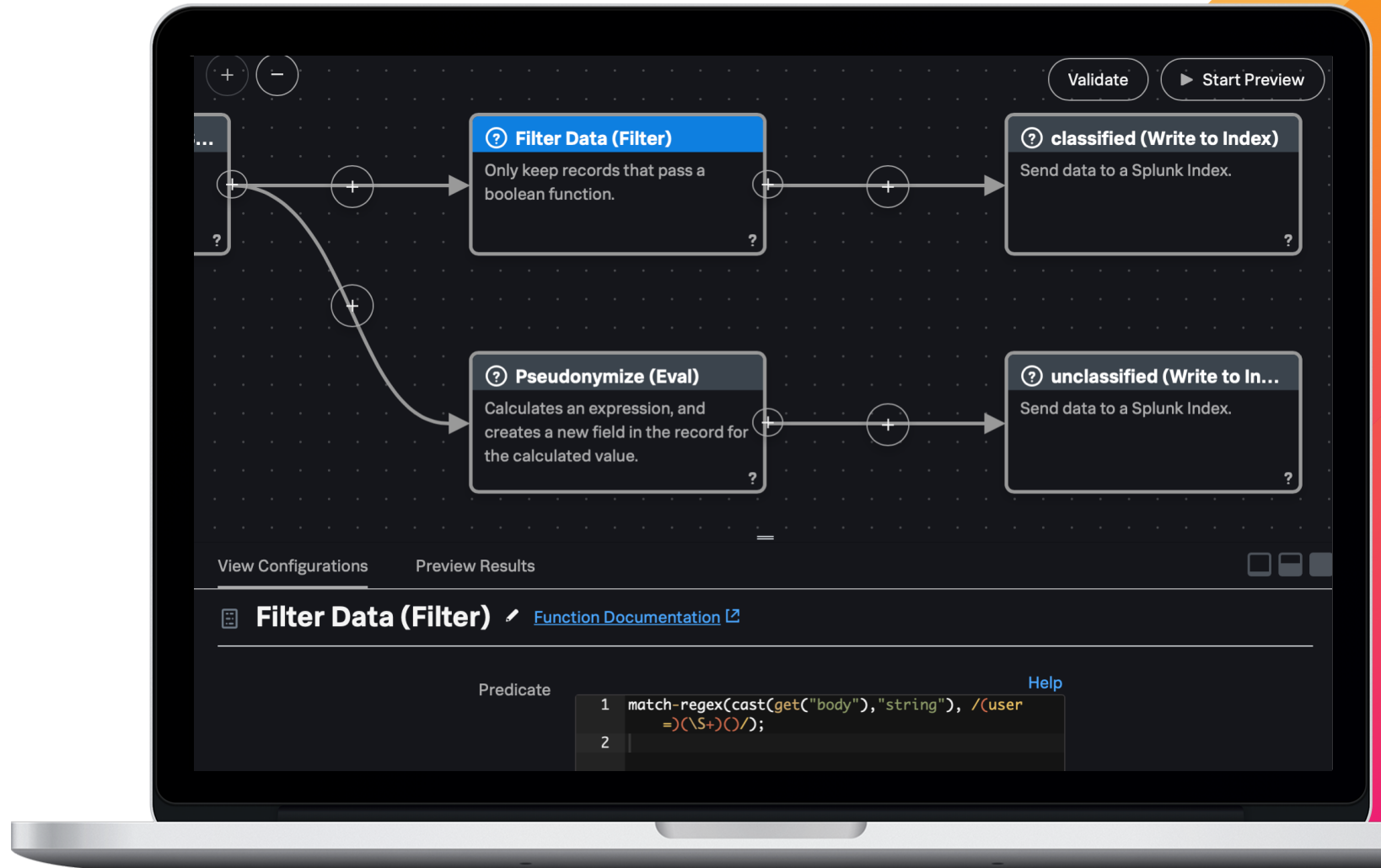
# Pseudonymize



# Anonymize (Multiple Categories)

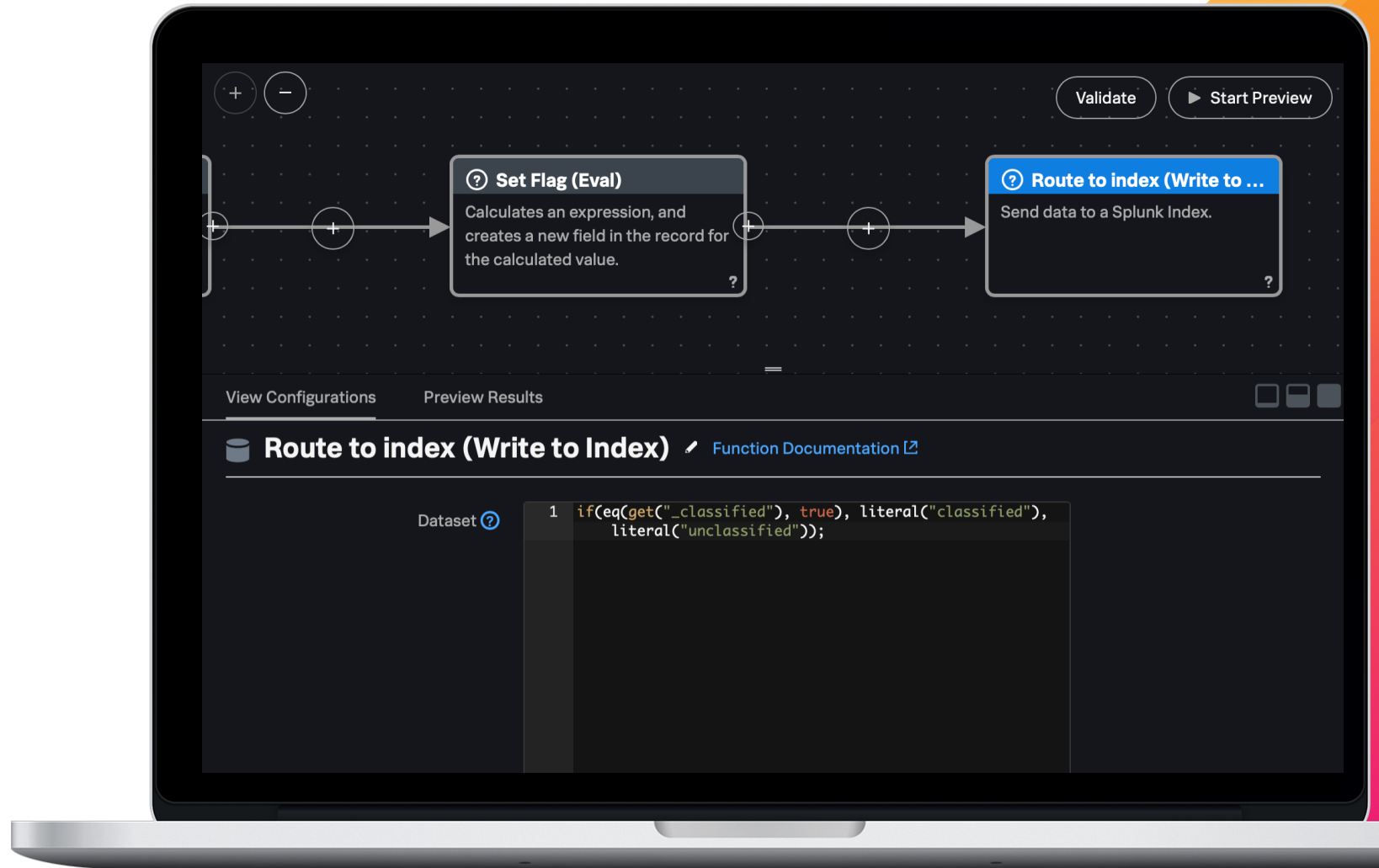


# Duplicate

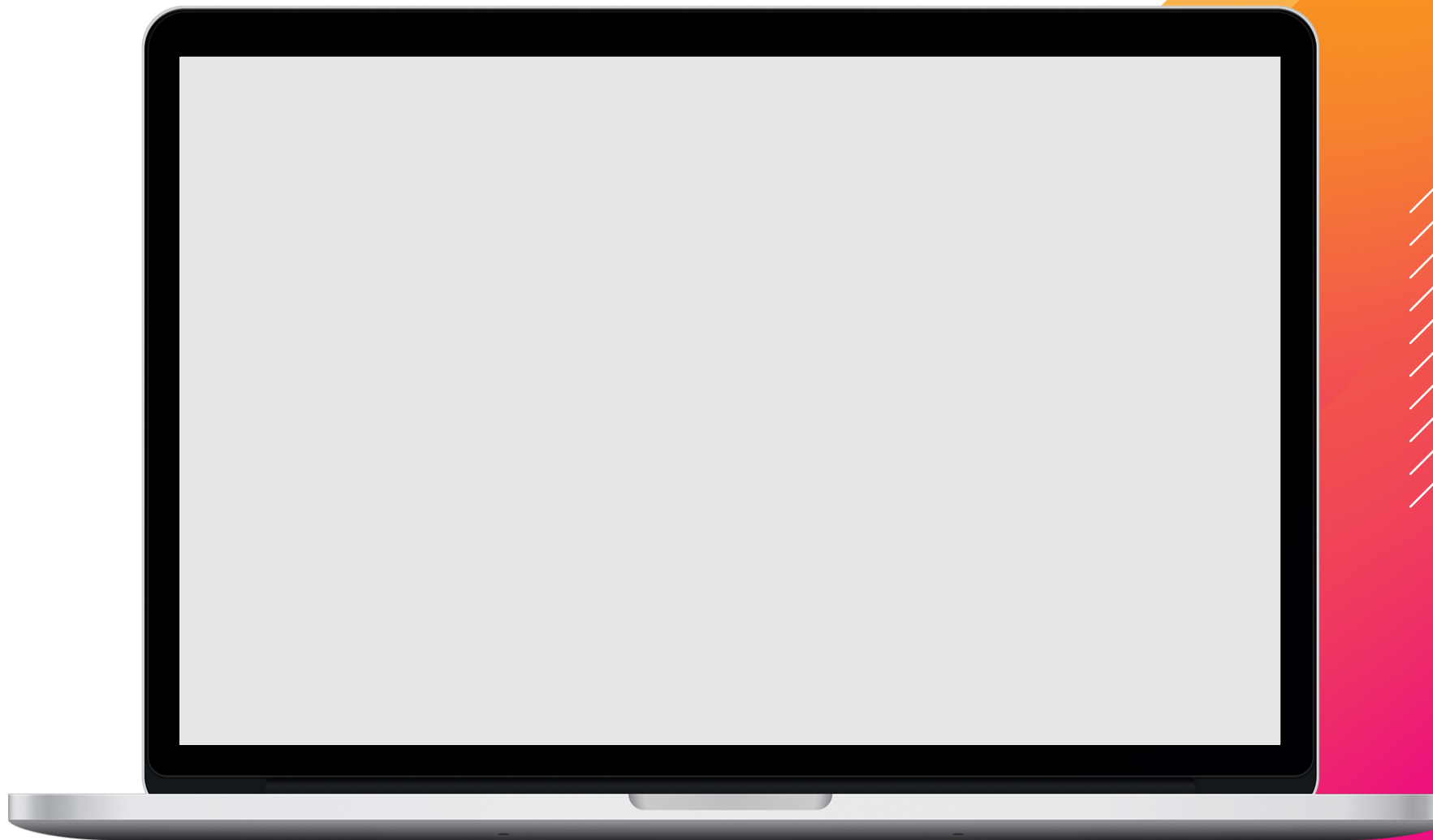




# Route to Index

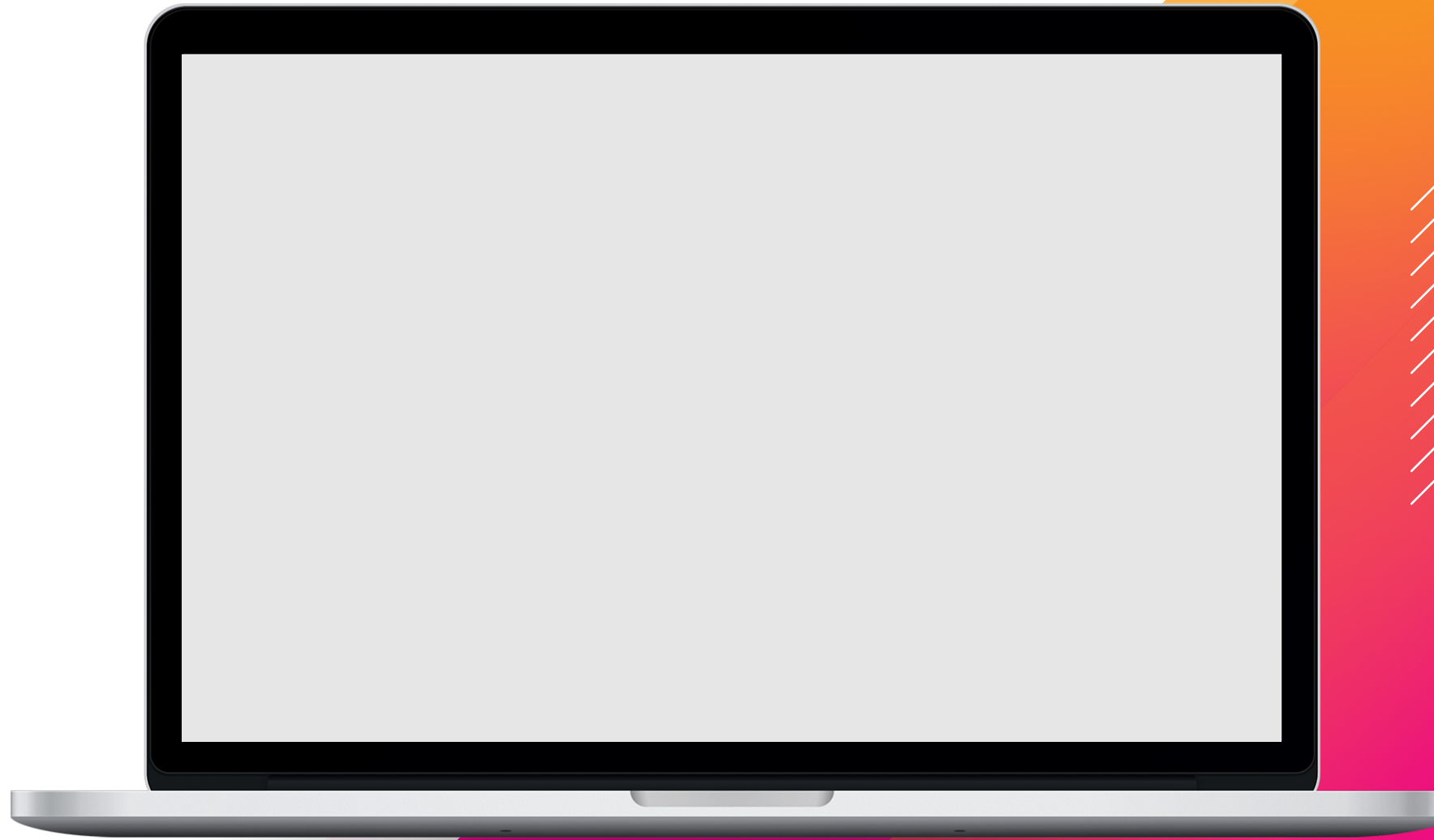


# Apply Own Function





# Enrich Data





# DSP as an Alerting and Action Engine

---

Learn how new functions in DSP can help Splunk real-time searching

# New Feature in DSP

Aggregate with Trigger Function

**If I see “X” more than “N” number of times,  
generate and index an event.**

**Example: If the number of HTTP 500 status  
code is > 10 in the last 5 minutes, generate  
a custom event in Splunk.**

# Aggregate with Trigger

A new function to help with real-time searches

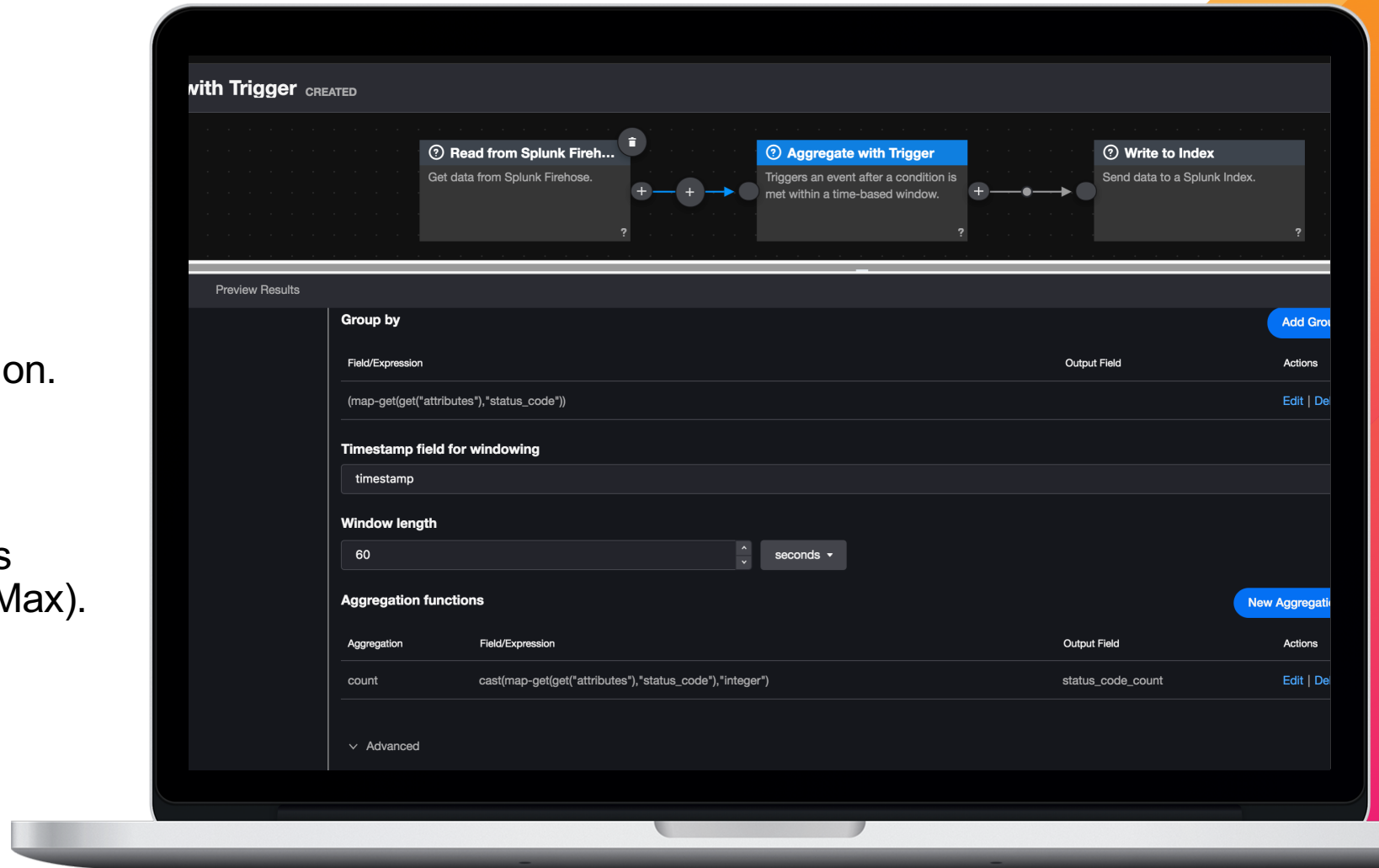
- New Function in DSP allows you to aggregate on the data pre-indexing while data is in flight.
- Unlike regular “Aggregate” function, this Function can fire while the aggregation window is still open if a condition is met.
- Real time searching can be greatly optimized by doing most of the heavy lifting work in DSP.
- Reduces the load on Splunk Indexers leading to a much more efficient system.



# Aggregate with Trigger

## Aggregations

- Group by fields to aggregate on.
- Define the Window Length.
- Define the Timestamp Field.
- Define Aggregation Functions (Sum, Count, Average, Min, Max).
- Define Windowing method (Tumbling vs Sliding)



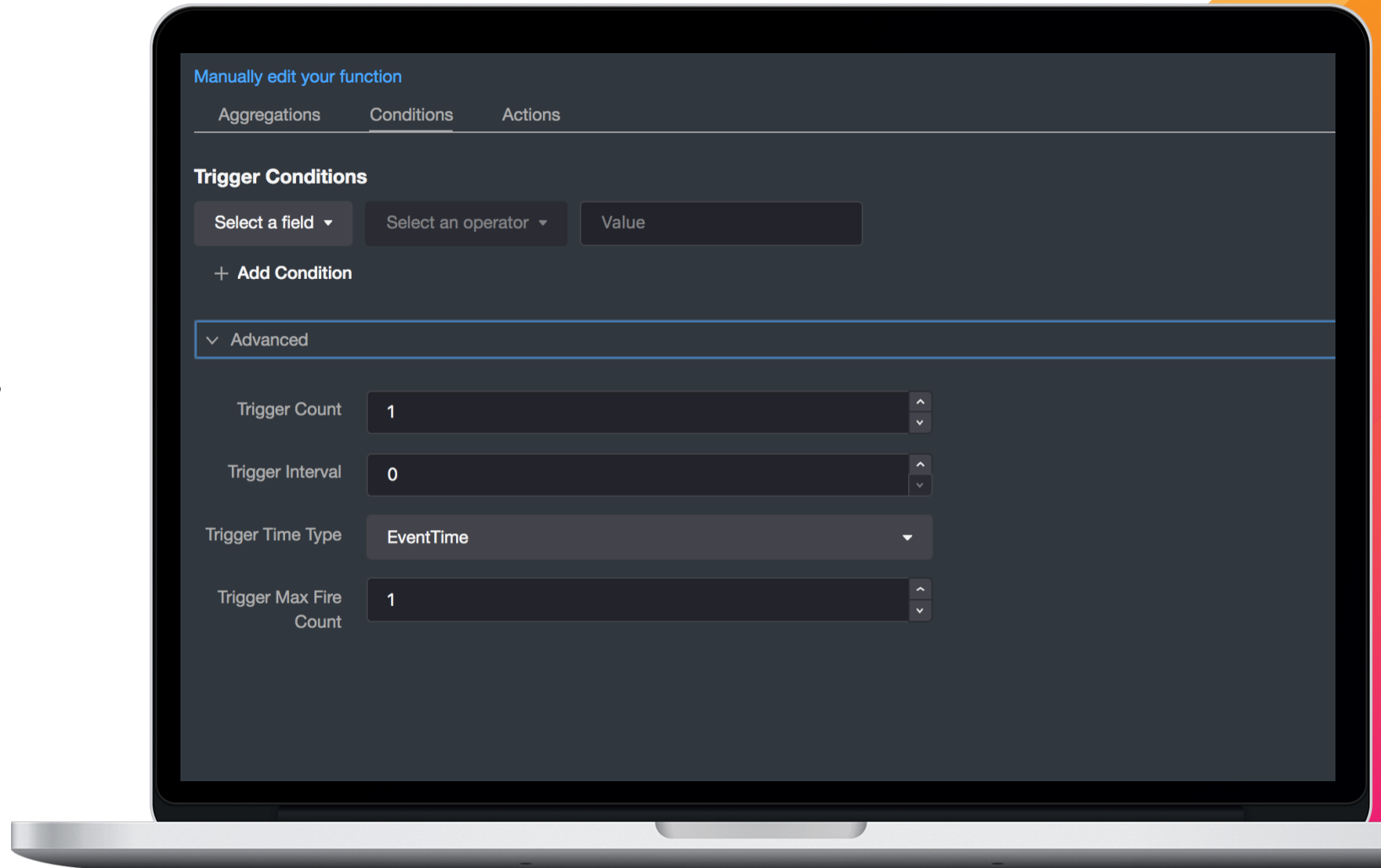
# Aggregate with Trigger

## Conditions

Define Trigger Conditions

Define Trigger Options:

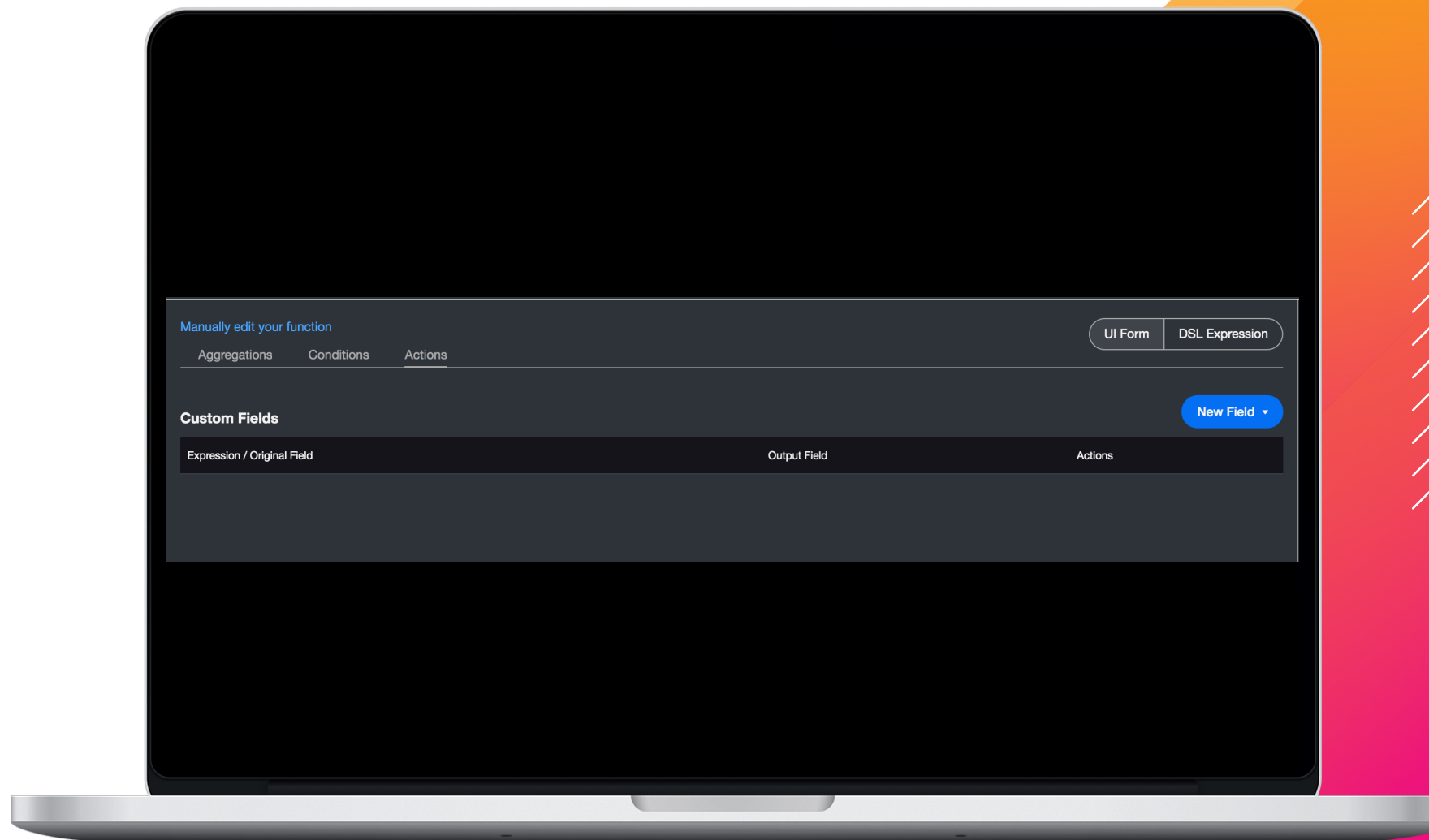
- Trigger Count
- Trigger Interval
- Trigger Time Type (Event Time vs. Processing Time)
- Trigger Maximum Fire Count



# Aggregate with Trigger

## Actions

Expression for what  
custom fields to use.





# Demo

# Key Takeaways

1. DSP offloads work from your Splunk Enterprise indexers
2. DSP can be used to transform incoming data according to your needs
3. DSP can be used as an alerting engine in conjunction with Splunk Enterprise to alert on events in real-time



# Q&A

---

Dirk Nitschke | Staff Sales Engineer  
Bashar Abdul Jawad | Principal Software Engineer



# Thank You!

Go to the .conf19 mobile app to

**RATE THIS SESSION**

