

Data Stream Processor

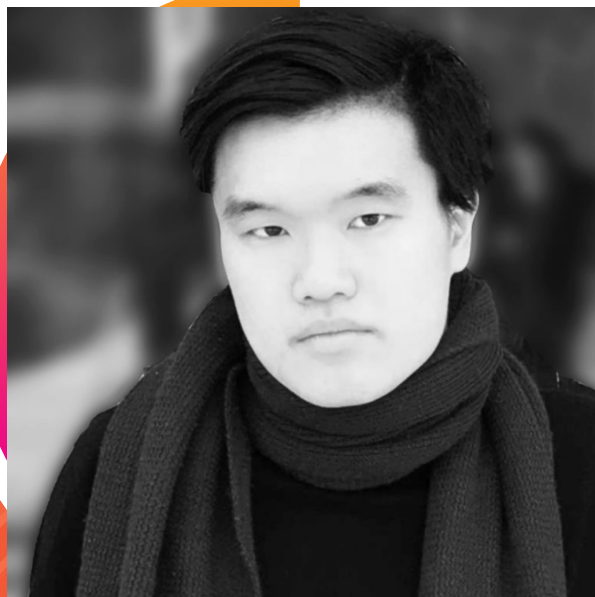
Architecture and SDKs

Max Feng | Software Engineer | Splunk

Sharon Xie | Senior Software Engineer | Splunk

October, 2019





Max Feng

Software Engineer | Splunk



Sharon Xie

Senior Software Engineer | Splunk

Forward-Looking Statements



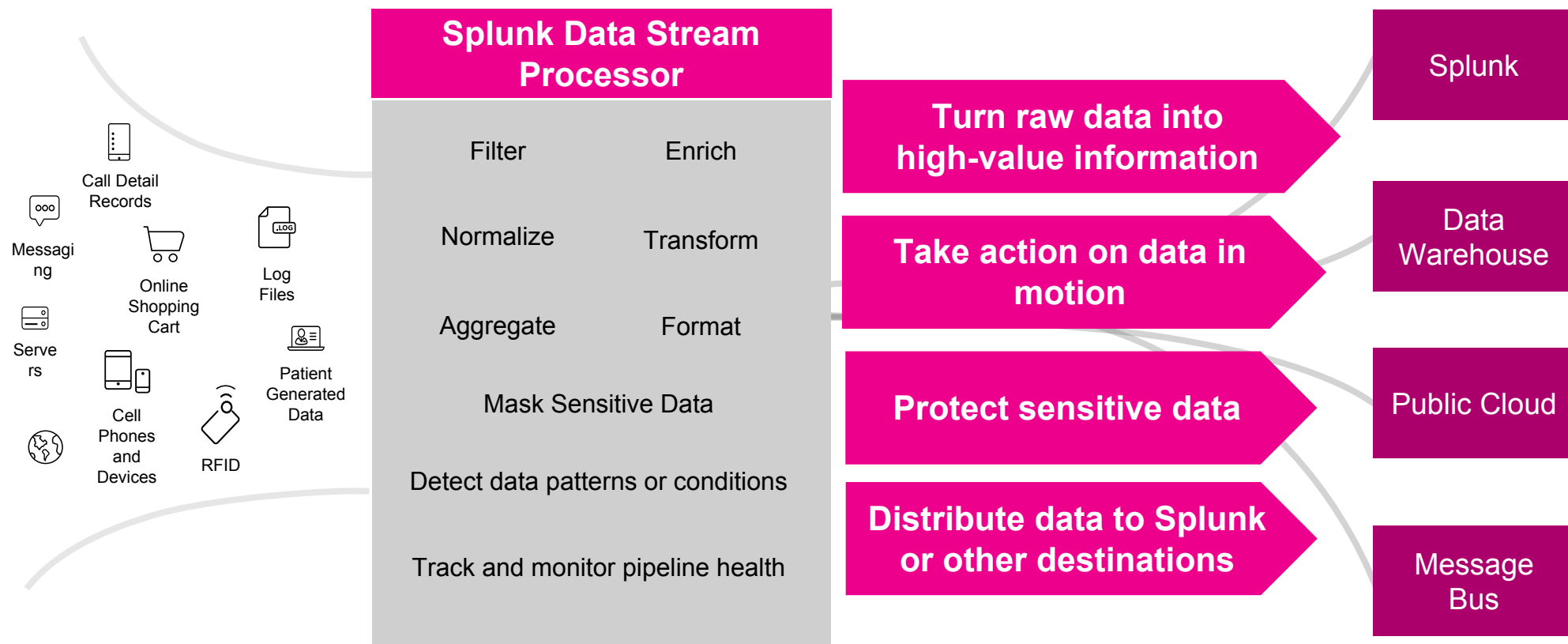
During the course of this presentation, we may make forward-looking statements regarding future events or plans of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results may differ materially. 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.

Splunk Data Stream Processor

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





Stream Processing

A primer

Agenda

1. Stream Processing Primer
2. Development Tools for Streaming
3. Splunk Data Stream Processor
 - Design goals
 - Core architecture
4. Reusability and Extensibility
5. Demo
6. Q&A



Everything is a stream

Existing Solutions

Introduction

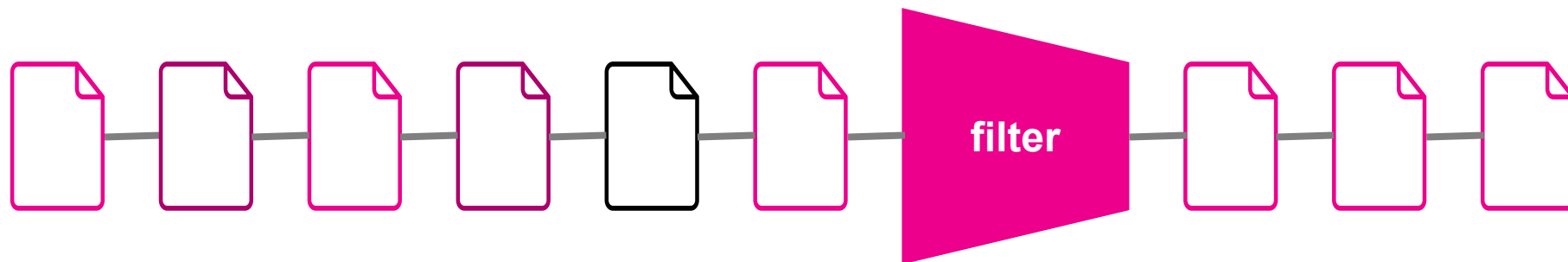
Frameworks

- Apache Flink
- Apache Spark
- Kafka Streams ...

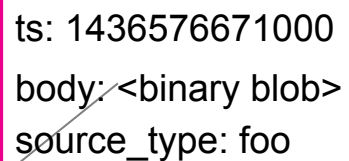
Operations on a stream

- Filter
- Extract
- Project
- Split
- Aggregate
- Join
- Model

Filter



Extract



```
ts: 1436576671000  
body: <binary blob>  
source_type: foo
```

extract

```
ts: 1436576671000  
body: <binary blob>  
source_type: foo  
attributes: {  
  ip: "127.0.0.1"  
  user_agent: "Mozilla/5.0"  
  user_id: "laura"  
  date: "[31/March/2016]"  
  request: "GET /index.html HTTP/1.0"  
  status_code: "200"  
  size: "2326"  
}
```

127.0.0.1 Mozilla/5.0 laura[31/Mar/2016] "GET /index.html HTTP/1.0" 200 2326

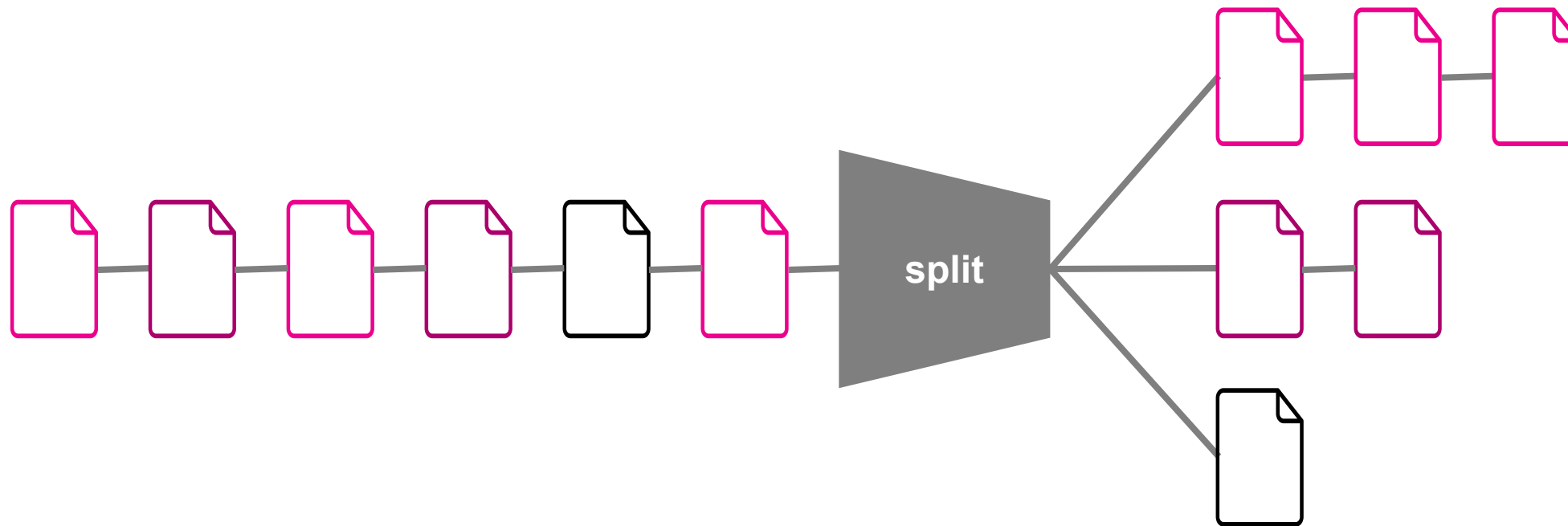
Project

```
ts: 1436576671000
body: <binary blob>
source_type: foo
attributes: {
  ip: "127.0.0.1"
  user_agent: "Mozilla/5.0"
  user_id: "laura"
  date: "[31/March/2016]"
  request: "GET /index.html HTTP/1.0"
  status_code: "200"
  size: "2326"
}
```

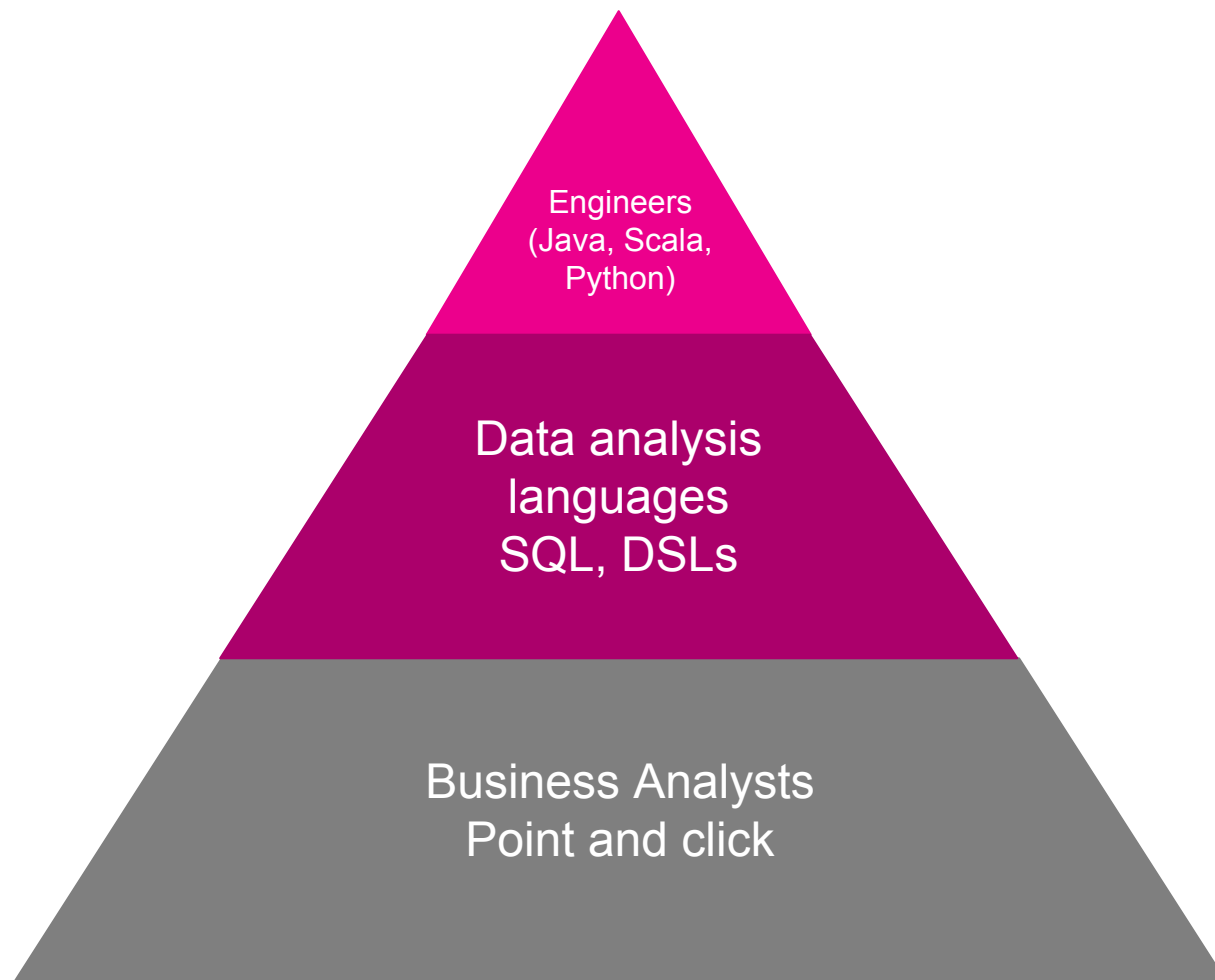
project

```
ts: 1459444413000
body: <binary blob>
source_type: foo
ip: "127.0.0.1"
user_agent: "Mozilla/5.0"
user_id: "laura"
request: "GET /index.html HTTP/1.0"
status_code: 200
size: 2326
```

Split



Development Tool Pyramid



```
DataStream<Tuple2<String, Long>> input = ...;  
input  
  .keyBy("productId")  
  .window(  
    TumblingEventTimeWindows.of(  
      Time.minutes(30)  
    )  
  )  
  .on("orderTime")  
  .aggregate(Aggregations.SUM, 0);
```

```
SELECT STREAM  
  TUMBLE_END(ordertime, INTERVAL '30' MINUTE, TIME '0:12') AS  
  orderTime, productId,  
  SUM(units) AS units  
FROM Orders  
GROUP BY TUMBLE(ordertime, INTERVAL '30' MINUTE, TIME '0:12'),  
productId;
```

Point and Click Stream Processing

The screenshot displays the Splunk Stream Processing interface. At the top, a pipeline is visualized with three blocks: **Read Data Source** (Get data from Splunk Firehose.), **Aggregate** (Applies aggregation functions, such as average, max, and count, on grouped records in a time window.), and **Output Data** (Send data to a Splunk Index.). Below the pipeline, the **Aggregate** block is selected, showing its configuration options. The interface includes a search bar for fields, a list of available fields (timestamp, nanos, id, host, source, source_type, kind), and a table for defining aggregation functions. The table has columns for Field/Expression, Output Field, and Actions. The configuration section includes a **Group by** section, a **Timestamp field for windowing** section, a **Window length** section (set to 60 seconds), and an **Aggregation functions** section.

Aggregate [Function Documentation](#)

Search Field

- # timestamp
- # nanos
- a id
- a host
- a source
- a source_type
- a kind

Manually edit your function

UI Form | DSL Expression

Group by [Add Groupby](#)

Field/Expression	Output Field	Actions
------------------	--------------	---------

Timestamp field for windowing

Field name or expression

Window length

60 seconds

Aggregation functions [New Aggregation](#)

Output fields

Key Takeaways

1. Stream processing frameworks are incredibly powerful.
2. The addressable audience increases with more accessible languages/tools.
3. Operational intelligence-style streaming tools are not as widely available.



Splunk Data Stream Processor

A solution

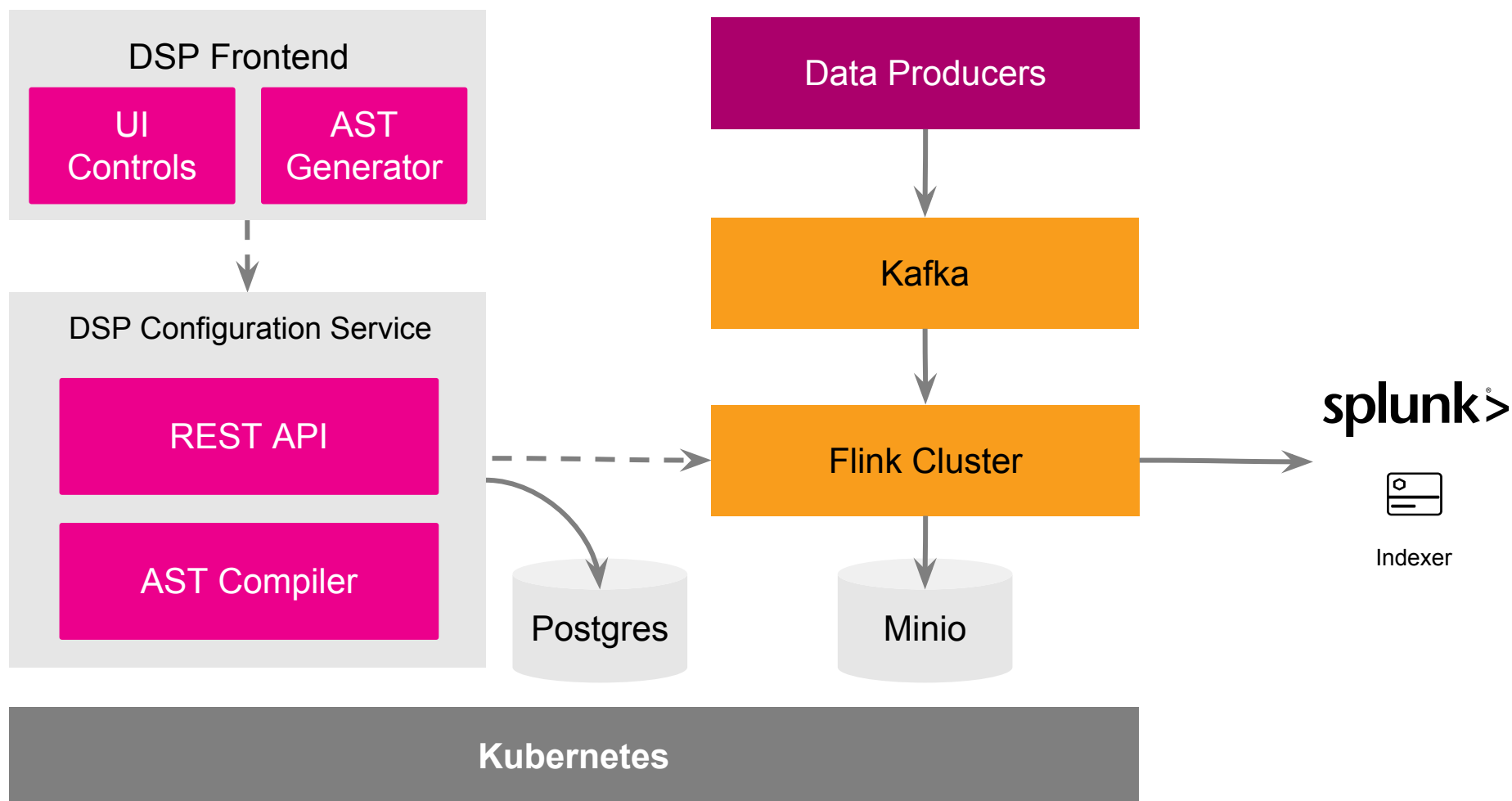
Business Requirements

Make stream processing usable without programming skills

While ensuring

- No data loss
- Fault tolerance
- High availability
- Easy scalability
- Observability

Overall System



Overall System

No data loss

- At least once delivery provided by Apache Flink and Apache Kafka

Fault tolerance

- States are preserved in persistent storage

High availability

- Clustered deployment managed by Kubernetes

Easy scalability

- Resource managed by Kubernetes

Observability

- Real-time and historical metrics
- System logs are available in Splunk

Our Solution

Make data stream processing usable without programming skills

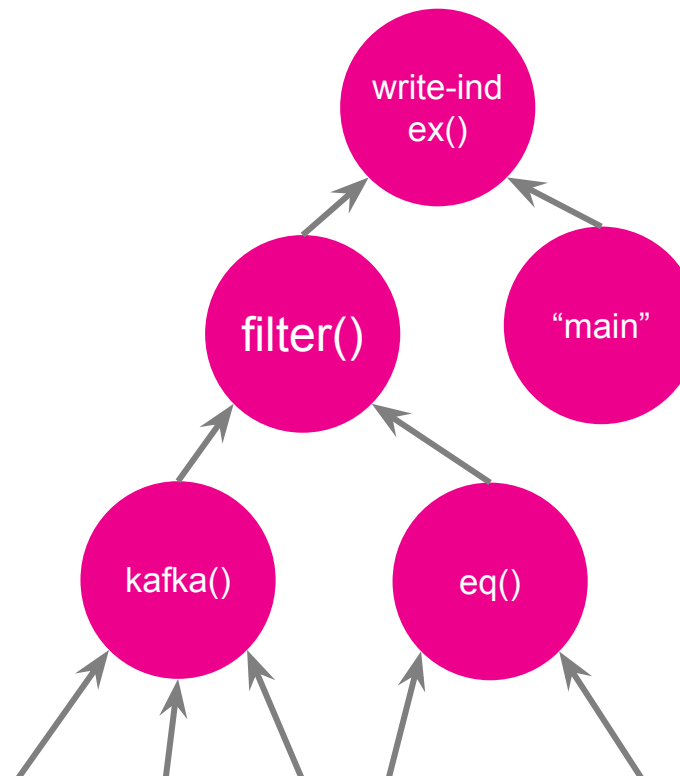
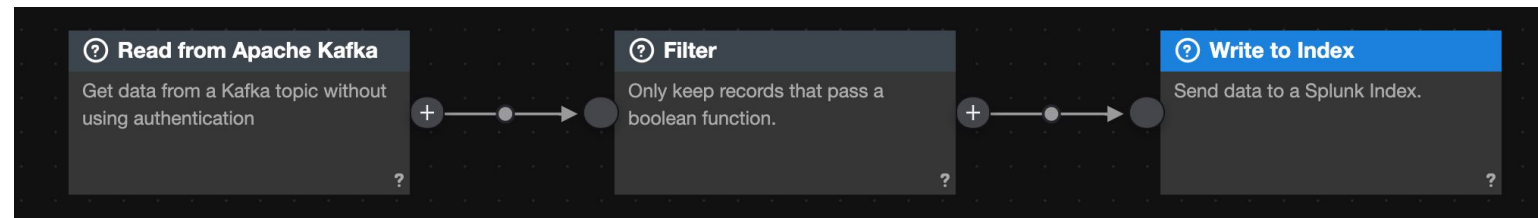
- Real-time visual authoring
- Real-time visual preview

How?

- Abstract Syntax Tree (AST)

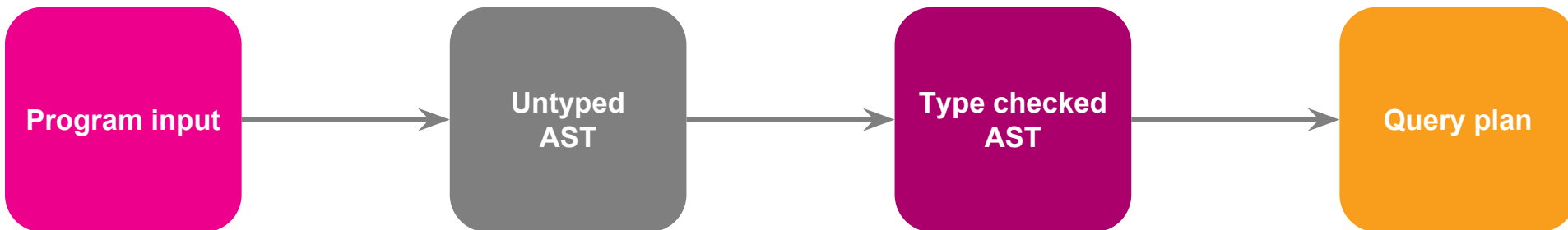
Real-time Visual Authoring

```
{
  nodes: [
    {
      id: "node-1",
      name: "kafka",
      attributes: { ... }
    },
    {
      id: "node-2",
      name: "filter",
      attributes: { ... }
    },
    {
      id: "node-3",
      name: "write-index",
      attributes: { ... }
    }
  ],
  edges: [
    {
      source: "node-1",
      target: "node-2"
    },
    {
      source: "node-2",
      target: "node-3"
    }
  ],
  rootNode: [ "node-1" ]
}
```

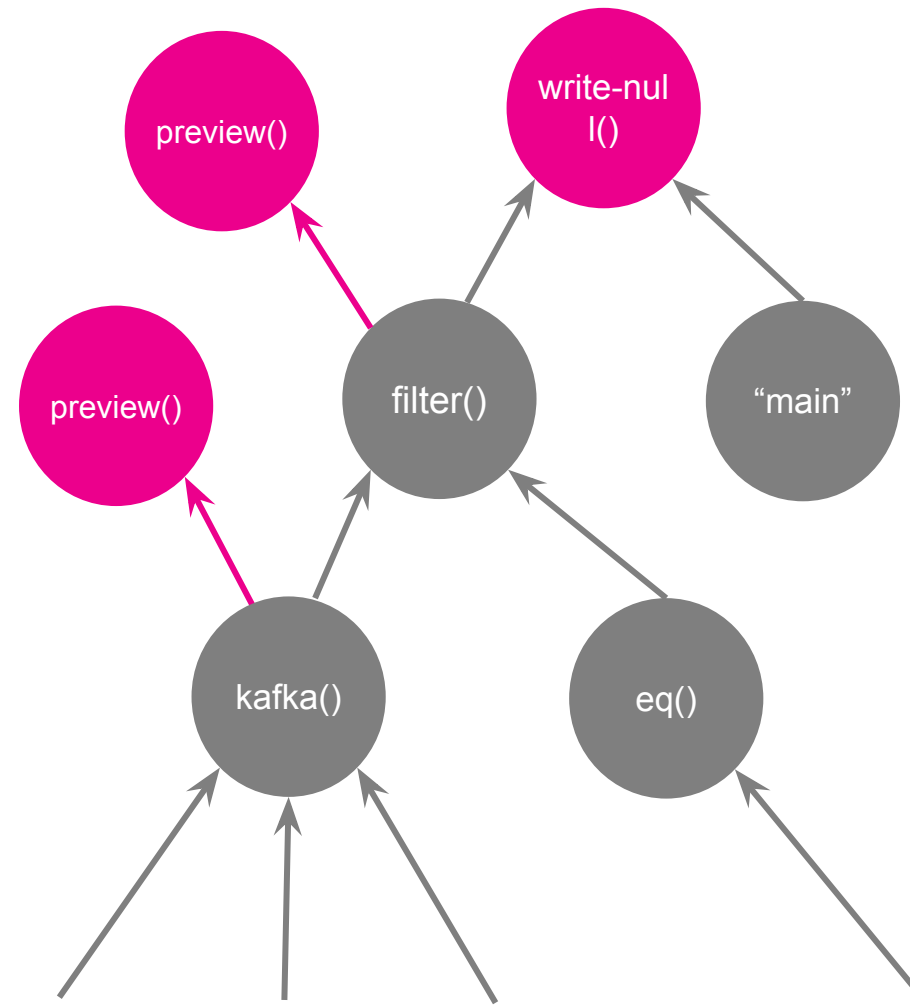
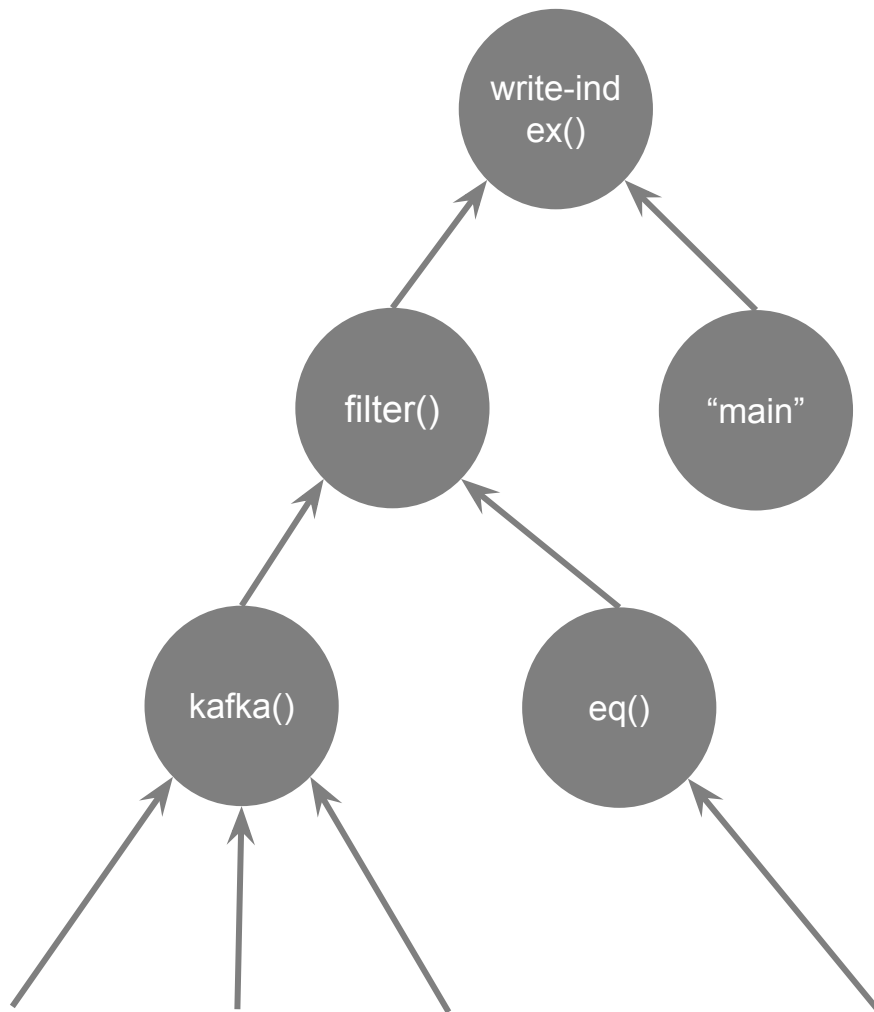


AST Lifecycle

Pipeline validation is done statically before running the job



Real-time Visual Preview





Reusability and Extensibility

A bonus



Group Functions

- A reusable, user-defined function containing any DSP expression
- Maps group function arguments to internal function arguments
- Usable anywhere built-in functions are, nest into other group functions

```
mask_ssn(input, field) => eval(input,
  as(if(
    match-regex(
      get(field),
      /d{3}-d{2}-d{4}/
    ),
    "XXX-XX-XXXX",
    get(field)
  ), field)
);
```

Templates

- A partially or fully configured pipeline
- Creating pipelines from templates pre-populates functions and arguments

Templates

Templates are partially or fully configured pipelines for specific use cases. You can use a template to jumpstart creating your pipeline.

i

NAME ^

ACTIONS

> Aggregate Buttercup Data	Edit	Delete	Clone
> AWS S3	Edit	Delete	Clone
> Buttercup Games Transaction Data	Edit	Delete	Clone

Developer SDK

What if the function you want isn't in DSP?

- Fancy machine learning functions
- Blockchain consensus processing

DSP provides a developer SDK for extensibility

- Clone the starter kit
- Write custom user-defined functions in Java, implementing our function interfaces
- Build an uber JAR and upload it to DSP via REST endpoints
- Use in any DSP pipeline!

Demo

Get Started Today!

Hardware Requirements

- Minimum Node Requirement
 - CPU: 8 core (16 recommended)
 - Memory: 64GB (128GB recommended)
 - Network: 10GBPS
 - Storage: 1TB
- Minimum 5 Node Cluster

Supported Data Sources

- Kafka, Kinesis, S3, CloudTrail, Event Hubs, REST APIs, Splunk (Universal Forwarder, Heavy Weight Forwarder, Http Event Collector)

Supported Data Sources

- Kafka, Kinesis
- Splunk



Q&A

Max Feng | Software Engineer | Splunk
Sharon Xie | Senior Software Engineer |
Splunk



splunk>

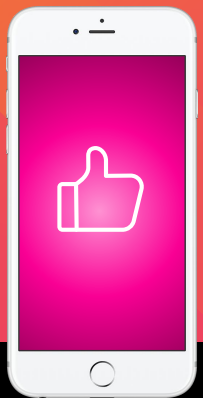
Thank

You



Go to the .conf19 mobile app to

RATE THIS SESSION



Other Data Stream Processor Sessions

1. FN1786 - Using Splunk Data Stream Processor for advanced stream management
2. FN1987 - Using Splunk Data Stream Processor as a streaming engine for Apache Kafka
3. FN2033 - Using Splunk Data Stream Processor as a Data Transformation, Altering, and Action Engine
4. FN2062 - Data Stream Processor: How to get the most out of your data!