

Patterns Of Streaming Applications

P
O
S
A

NETFLIX

| Monal Daxini
11/6/2018

 @monaldax

Profile



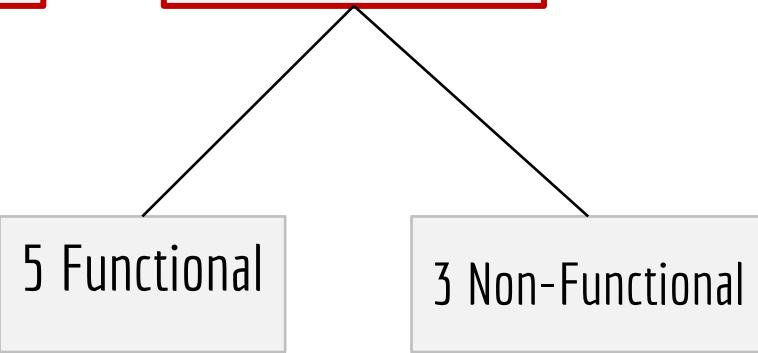
- 4+ years building stream processing platform at Netflix
 - Drove technical vision, roadmap, led implementation
- 17+ years building distributed systems

Structure Of The Talk

Stream
Processing ?

Set The Stage

8 Patterns





Disclaimer

Inspired by True Events encountered building and operating a Stream Processing platform, and use cases that are in production or in ideation phase in the cloud.

Some code and identifying details have been changed, artistic liberties have been taken, to protect the privacy of streaming applications, and for sharing the know-how. Some use cases may have been simplified.

Stream Processing? | Processing Data-In-Motion

NETFLIX ORIGINAL

STRANGER THINGS

95% Match 2016 1 Season 4K Ultra HD 5.1

When a young boy vanishes, a small town uncovers a mystery involving secret experiments, terrifying supernatural forces and one strange little girl.

Winona Ryder, David Harbour, Matthew Modine
TV Shows, TV Sci-Fi & Fantasy, Teen TV Shows



Popular on Netflix

NETFLIX

STRANGER THINGS

NETFLIX

okja

NETFLIX

GLOW

NETFLIX

13TH

NETFLIX

Master of None



Recently Watched

NETFLIX

CLUB de CUERVOS

NETFLIX

HOUSE of CARDS

NETFLIX THE RANCH

NETFLIX

THE CROWN

NETFLIX

CHEF'S TABLE

Lower Latency Analytics

Trending Now



Golden Globe Award-winning TV Shows >



Law & Order: Special Victims Unit

90% Match 2015 [TV-14] 3 Seasons

Welcome to the Special Victims Unit. Where the crème de la crème rain justice on the scum of the earth.

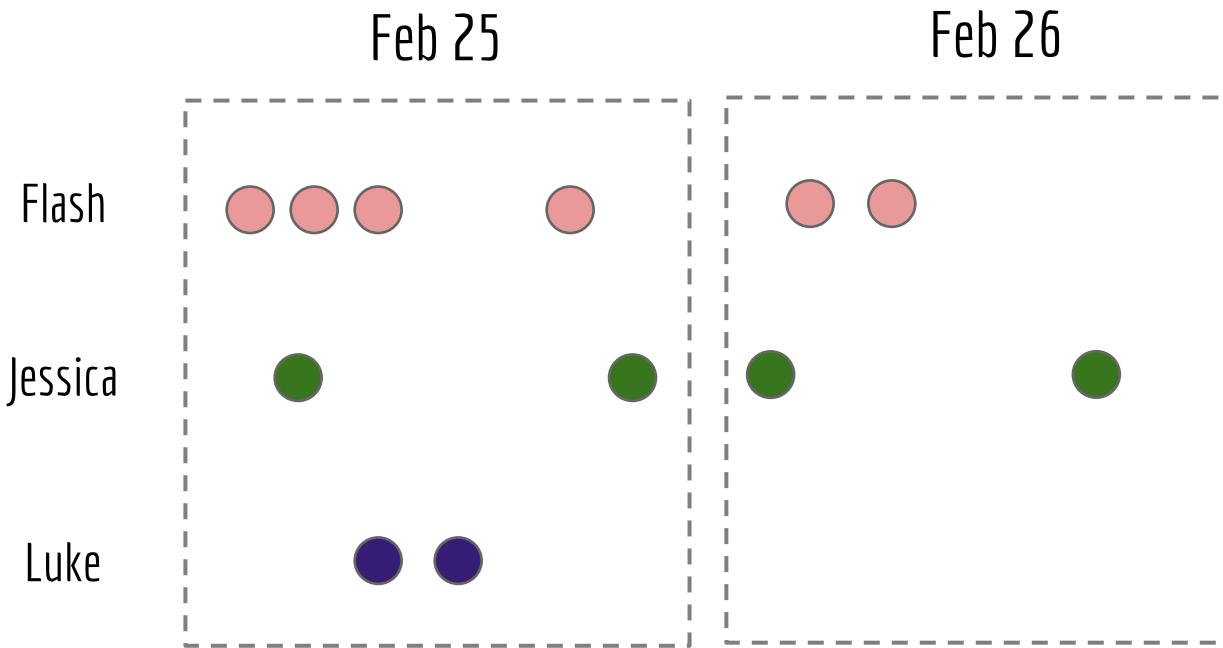
▼

A detailed view of a media card for 'Law & Order: Special Victims Unit'. It features a large play button icon in the center. Below it is the show's title and a summary. To the right are three circular icons with symbols: a thumbs up, a thumbs down, and a plus sign. At the bottom is a downward-pointing arrow.

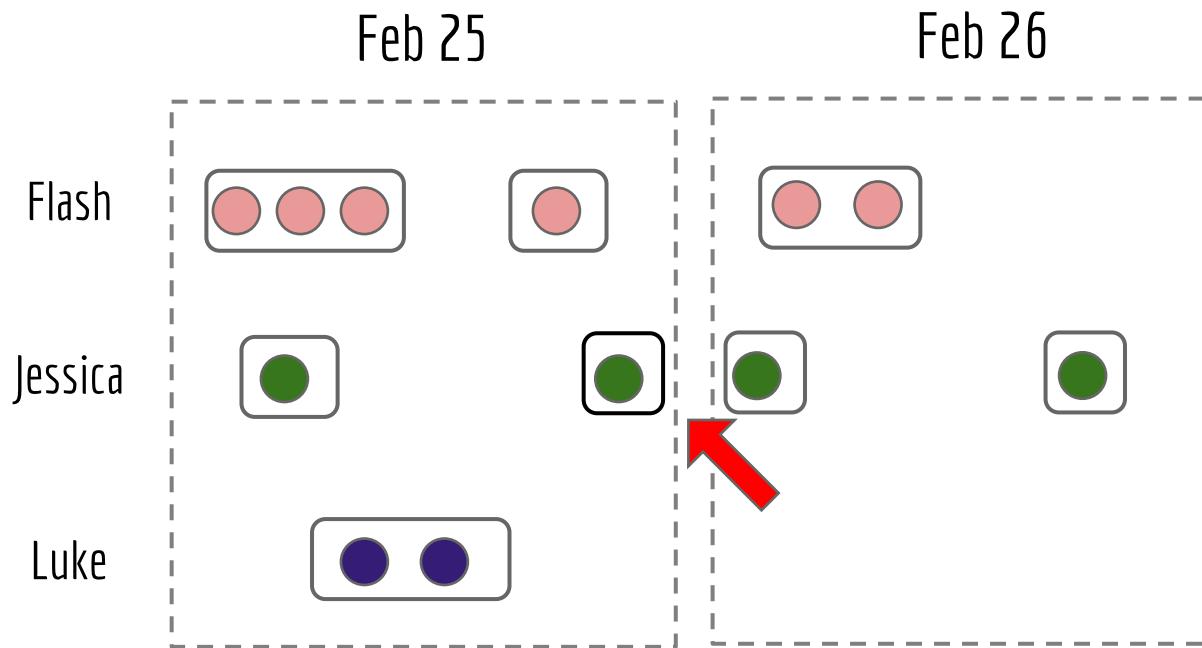
TV Shows



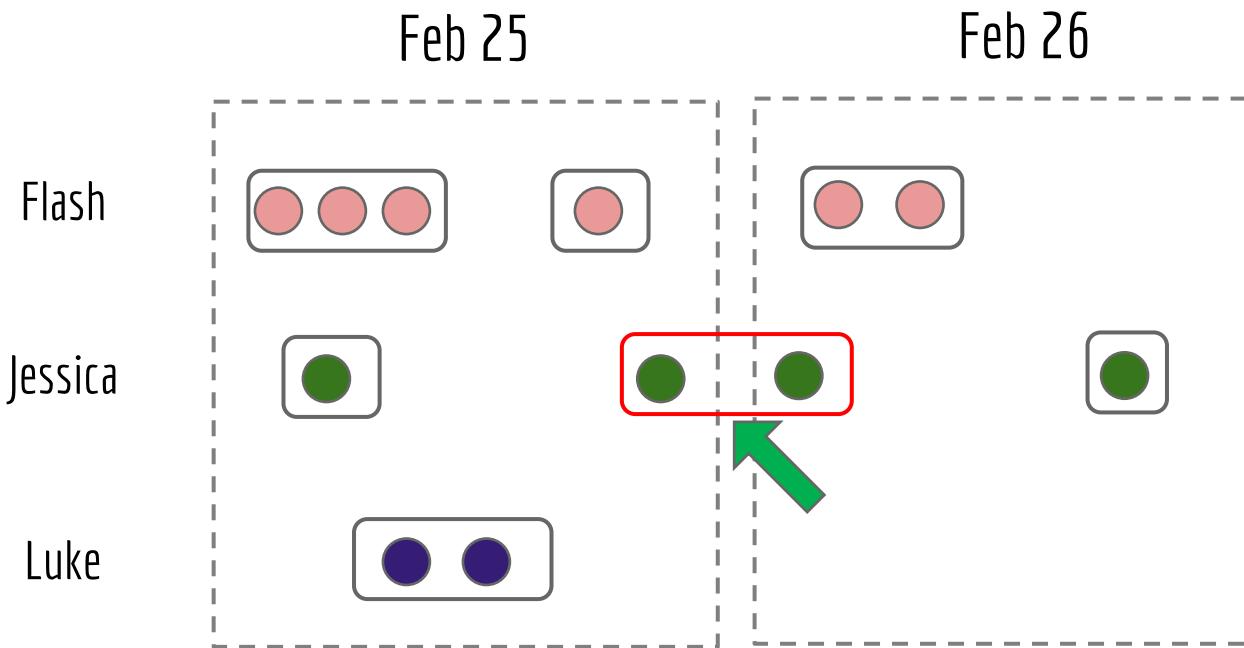
User Activity Stream - Batched



Sessions - Batched User Activity Stream



Correct Session - Batched User Activity Stream



Stream Processing Natural For User Activity Stream Sessions

Flash



Jessica



Luke



Why Stream Processing?

1. Low latency insights and analytics
2. Process unbounded data sets
3. ETL as data arrives
4. Ad-hoc analytics and *Event driven applications*

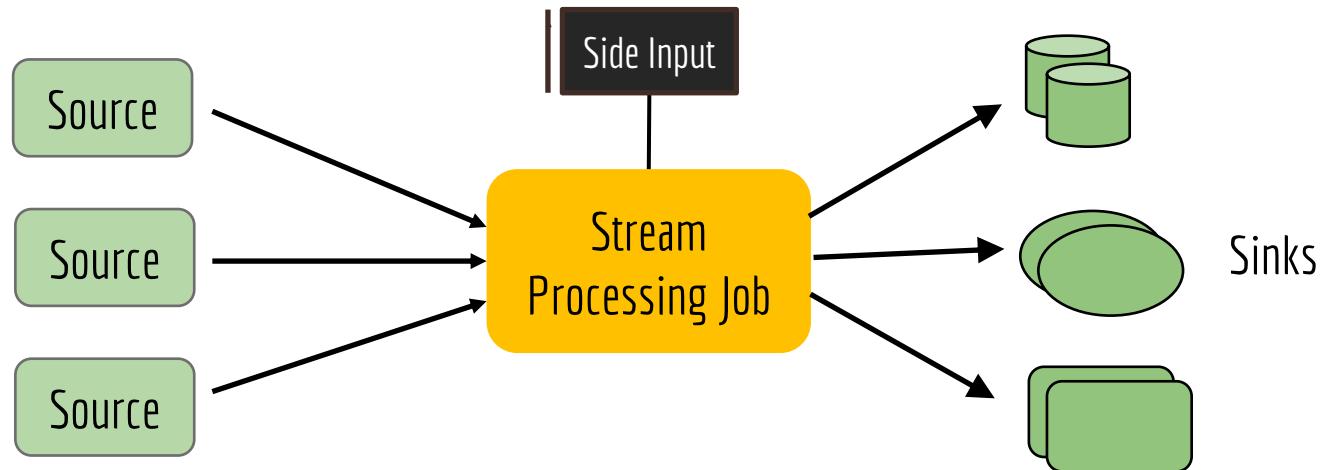
Set The Stage

| Architecture & Flink

Stream Processing App Architecture Blueprint



Stream Processing App Architecture Blueprint



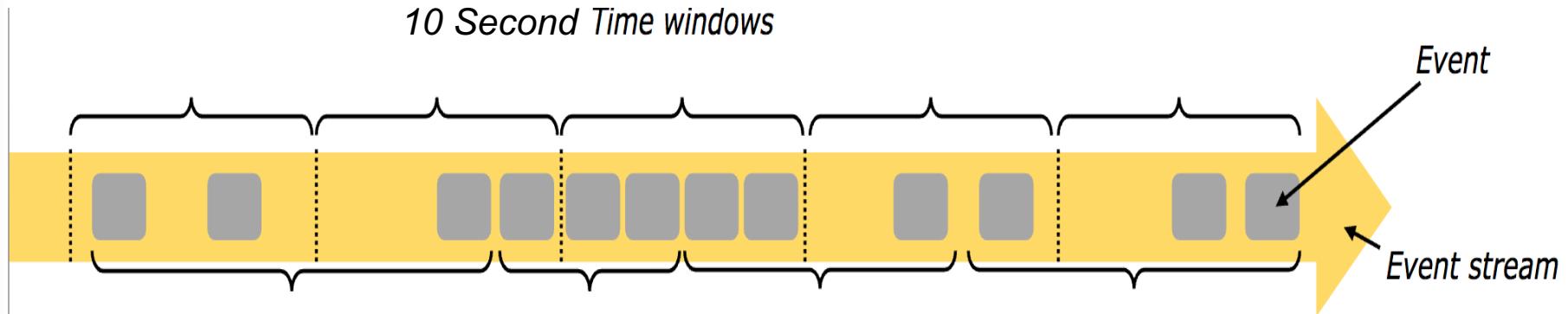


Why Flink?

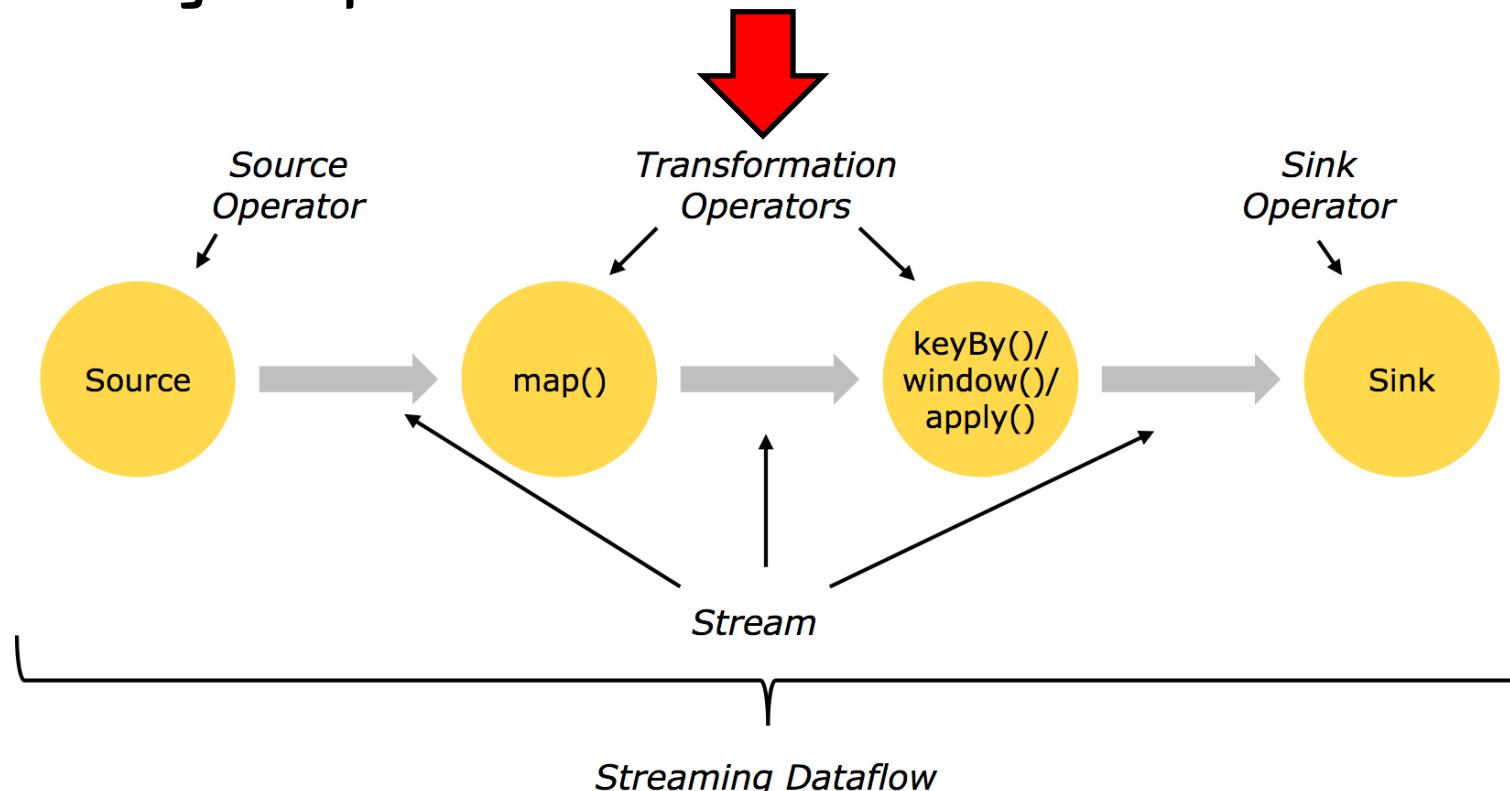
Flink Programs Are Streaming Dataflows - Streams And Transformation Operators

```
DataStream<String> lines = env.addSource(  
    new FlinkKafkaConsumer<>(...)); } Source  
  
DataStream<Event> events = lines.map((line) -> parse(line)); } Transformation  
  
DataStream<Statistics> stats = events  
    .keyBy("id")  
    .timeWindow(Time.seconds(10))  
    .apply(new MyWindowAggregationFunction()); } Transformation  
  
stats.addSink(new RollingSink(path)); } Sink
```

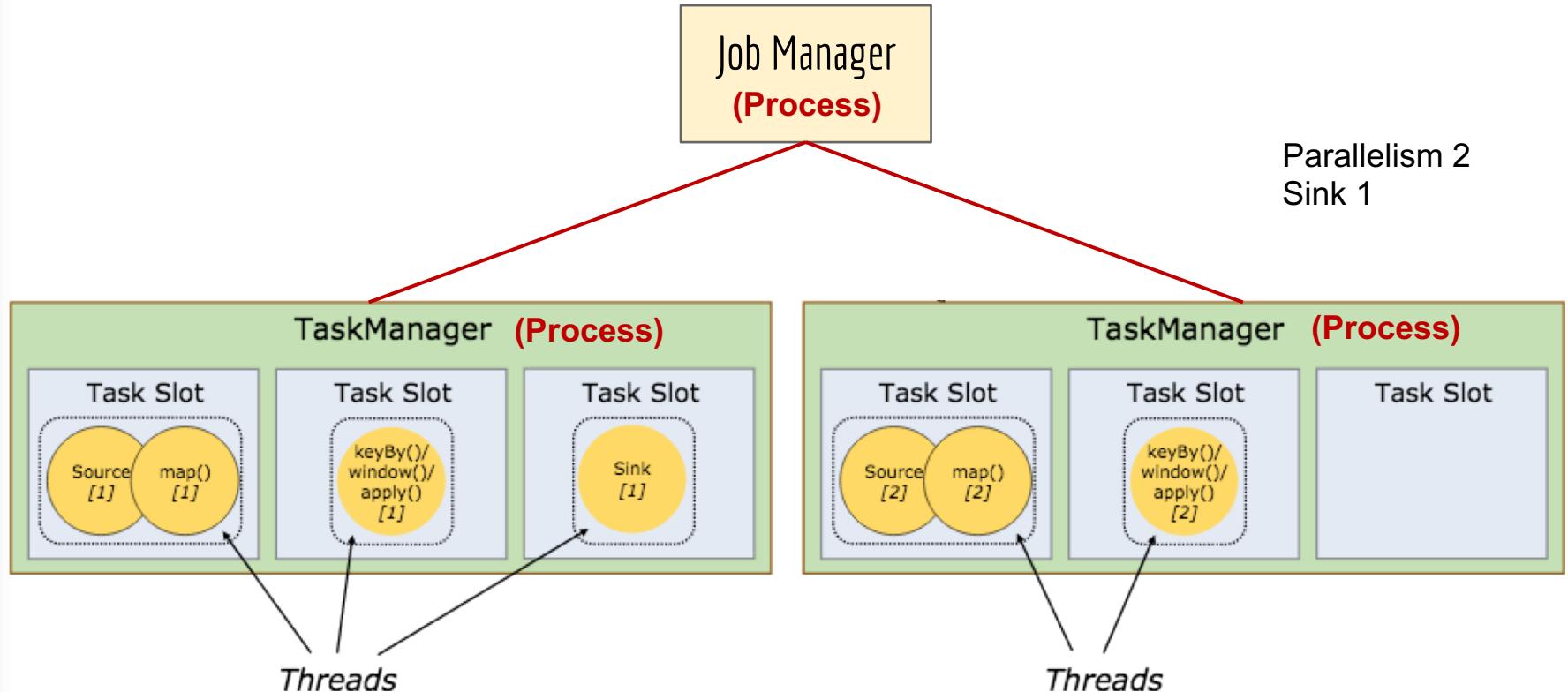
Streams And Transformation Operators - Windowing



Streaming Dataflow DAG



Scalable Automatic Scheduling Of Operations



Flexible Deployment

Bare Metal



VM / Cloud



Containers

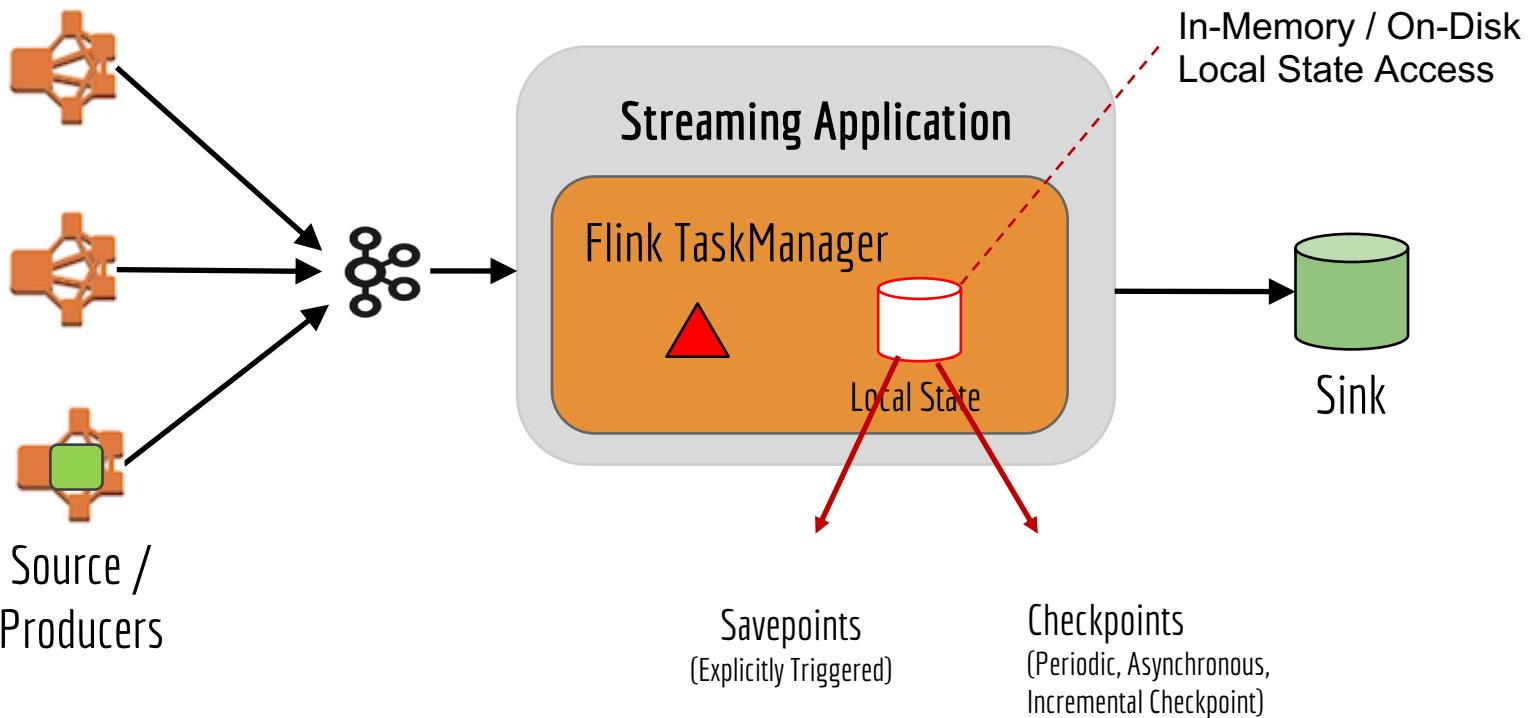


Stateless Stream Processing

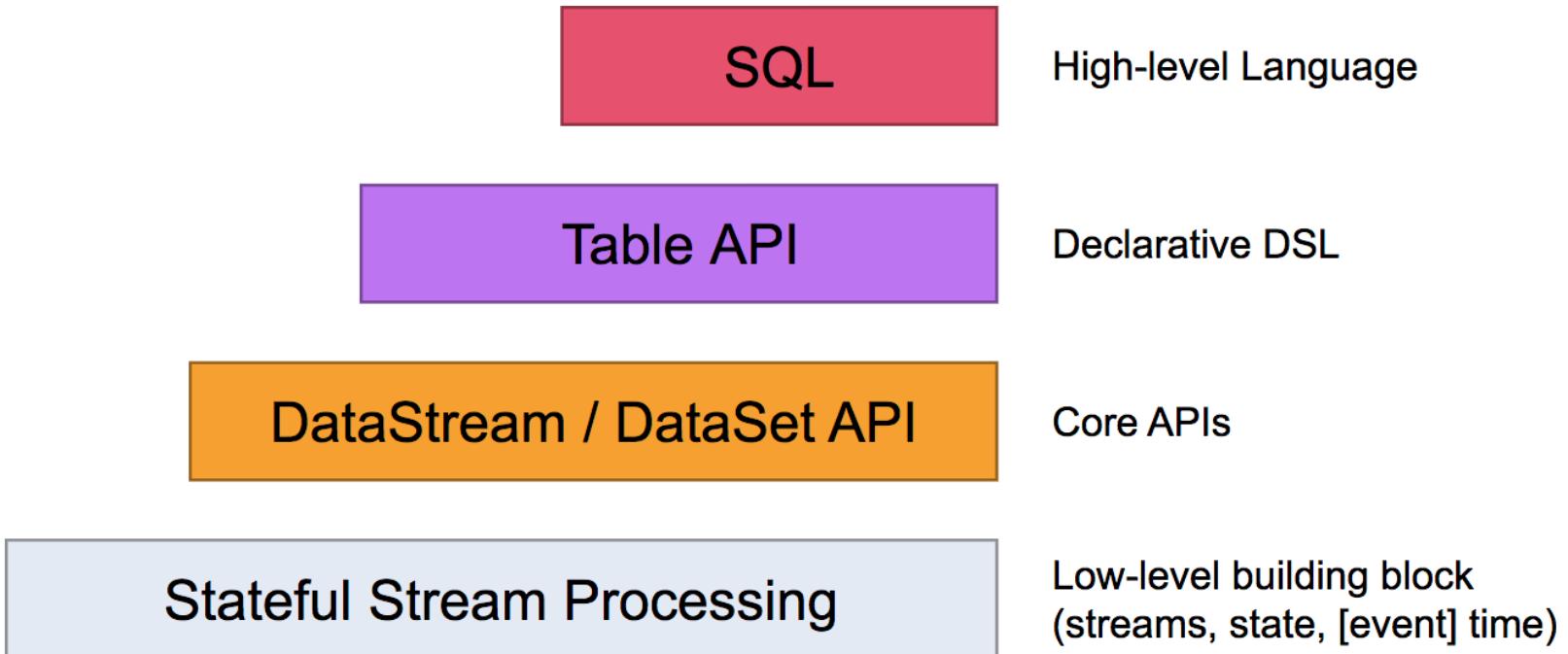
No state maintained across events



Fault-tolerant Processing - Stateful Processing



Levels Of API Abstraction In Flink



Source: Flink Documentation

Describing Patterns

Describing Design Patterns

- Use Case / Motivation
- Pattern
- Code Snippet & Deployment mechanism
- Related Pattern, if any

Patterns

| Functional

1. Configurable Router

NETFLIX ORIGINAL

STRANGER THINGS

95% Match 2016 1 Season 4K Ultra HD 5.1

When a young boy vanishes, a small town uncovers a mystery involving secret experiments, terrifying supernatural forces and one strange little girl.

Winona Ryder, David Harbour, Matthew Modine
TV Shows, TV Sci-Fi & Fantasy, Teen TV Shows



Popular on Netflix

NETFLIX

STRANGER THINGS

NETFLIX

okja

NETFLIX

GLOW

NETFLIX

13TH

NETFLIX

Master of None



Recently Watched

NETFLIX

CLUB de CUERVOS

NETFLIX

HOUSE of CARDS

NETFLIX

THE RANCH

NETFLIX

THE CROWN

NETFLIX

CHEF'S TABLE

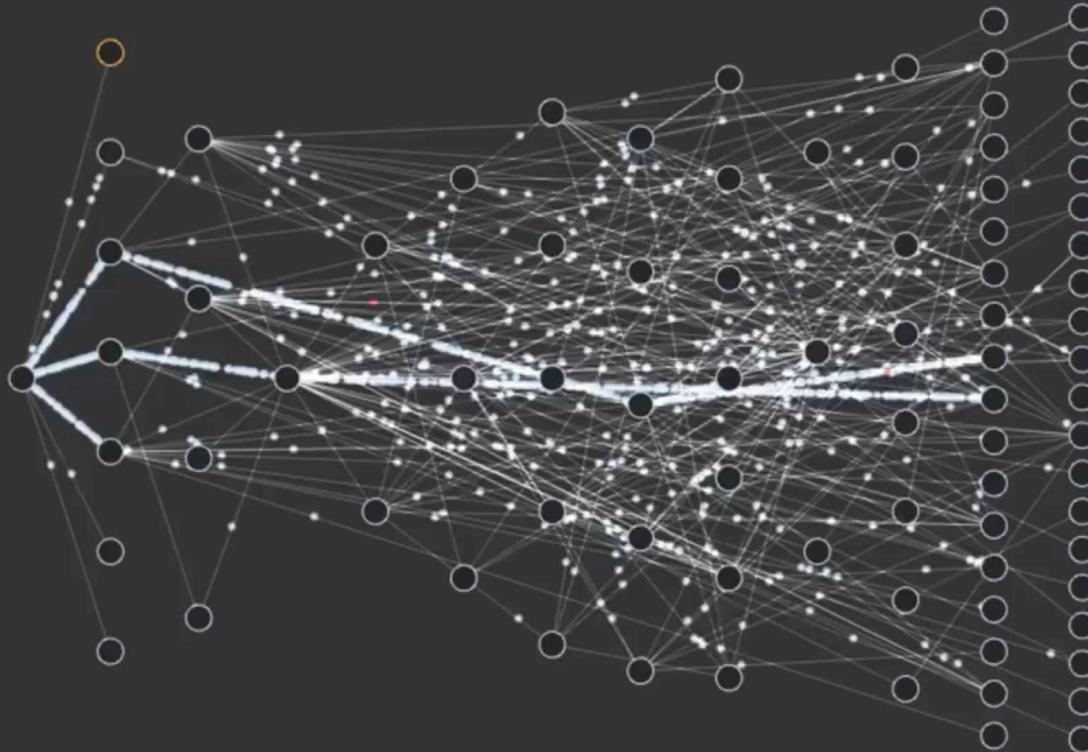
Service Traffic Map / us-east-1

200 services / 116 filtered (show)

Locate Service



Filters ▾ Display ▾



1.1 Use Case / Motivation - Ingest Pipelines

- Create ingest pipelines for different event streams **declaratively**
- Route events to data warehouse, data stores for analytics
 - With at-least-once semantics
- Streaming ETL - Allow declarative filtering and projection



1.1 Keystone Pipeline - A Self-serve Product



- SERVERLESS
- Turnkey - ready to use
- 100% in the cloud
- No code, Managed Code & Operations

1.1 UI To Provision 1 Data Stream, A Filter, & 3 Sinks

Stream Name	Owner	Description
cleverated Logging	cleverated@netflix.com	generated Logging

PROD.eu-west-1 PROVISIONED + Output + Filter + Projection Update trash

Dec 01, 2016 04:00 PM

The diagram illustrates the data flow for the stream 'cleverated Logging'. It starts with 'Producers' (represented by a blue circle) connected to 'Keystone' (blue circle). From Keystone, the flow splits into three parallel paths, each enclosed in a red dashed box:

- Elasticsearch:** Represented by a blue circle with a checkered background and a magnifying glass icon.
- Hive:** Represented by a blue circle with a yellow bee icon and the word 'HIVE'.
- K2:** Represented by a blue circle with a black and white checkered background and a cluster of nodes icon.

A 'Filter' (blue circle) is positioned between Keystone and the Elasticsearch sink. The entire stream structure is timestamped at Dec 01, 2016 04:00 PM.

1.1 Optional Filter & Projection (Out of the box)



Filter

Links ▾ Actions ▾ X

Attach new Outputs to this Filter/Projection using the "Stream Actions" menu, or drag-and-drop existing Outputs onto this node. Refer to the [XPathFilter syntax](#) reference for help defining your Filter expression.

Filter: Include only events that match this XPath expression

Filter XPath Expression

```
xpath("source") = "edx"
```

Projection: Include/Exclude specific fields from each event

Projection Behavior

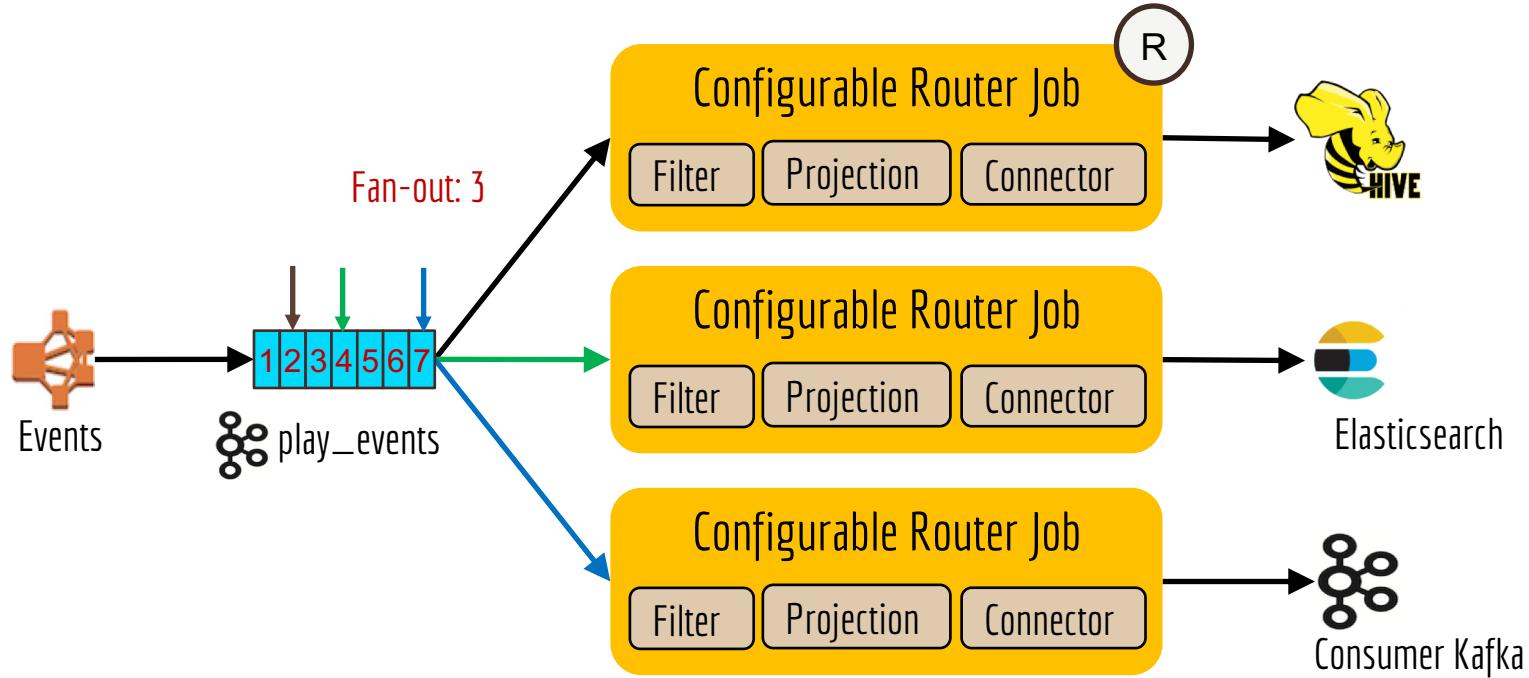
Include only selected fields

Exclude selected fields

Fields to include/exclude

Enter a comma-separated list of top-level Fields to include/exclude from the message

1.1 Provision 1 Kafka Topic, 3 Configurable Router Jobs



1.1 Keystone Pipeline Scale



- Up to 1 trillion new events / day
- Peak: **12M** events / sec, **36 GB** / sec
- ~4 PB of data transported / day
- ~2000 Router Jobs / 10,000 containers

1.1 Pattern: Configurable Isolated Router



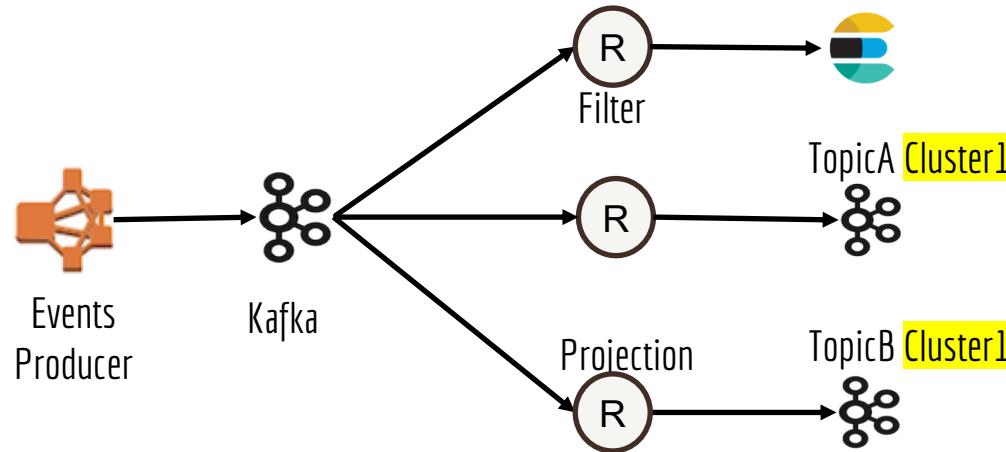
1.1 Code Snippet: Configurable Isolated Router

```
val kafkaSource = getSourceBuilder.fromKafka("topic1").build()  
  
val selectedSink = getSinkBuilder()  
    .toSelector(sinkName).declareWith("kafkasink", kafkaSink)  
    .or("s3sink", s3Sink).or("essink", esSink).or("nullsink", nullSink).build();  
  
kafkaSource  
    .filter(KeystoneFilterFunction).map(KeystoneProjectionFunction)  
    .addSink(selectedSink)
```

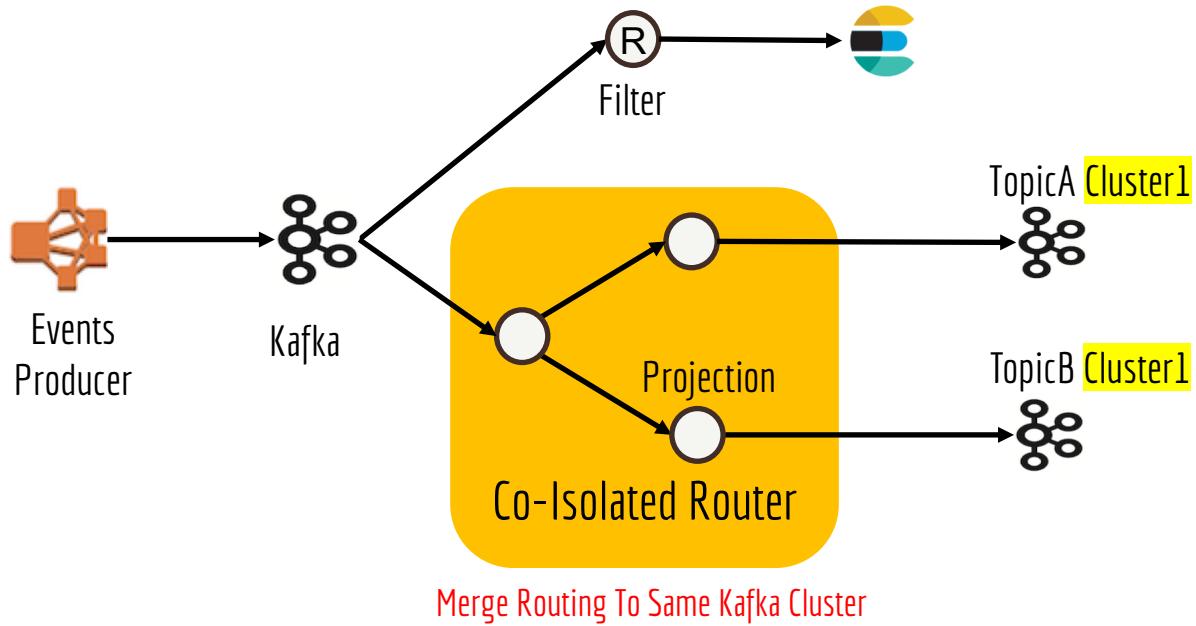
No User Code

1.2 Use Case / Motivation - Ingest large streams with high fan-out Efficiently

- Popular stream / topic has high fan-out factor
 - Requires large Kafka Clusters, expensive



1.2 Pattern: Configurable Co-Isolated Router



1.2 Code Snippet: Configurable Co-Isolated Router

```
ui_A_Clicks_KafkaSource  
.filter(filter)  
.map(projection)  
.map(outputConverter)  
.addSink(kafkaSinkA_Topic1)
```

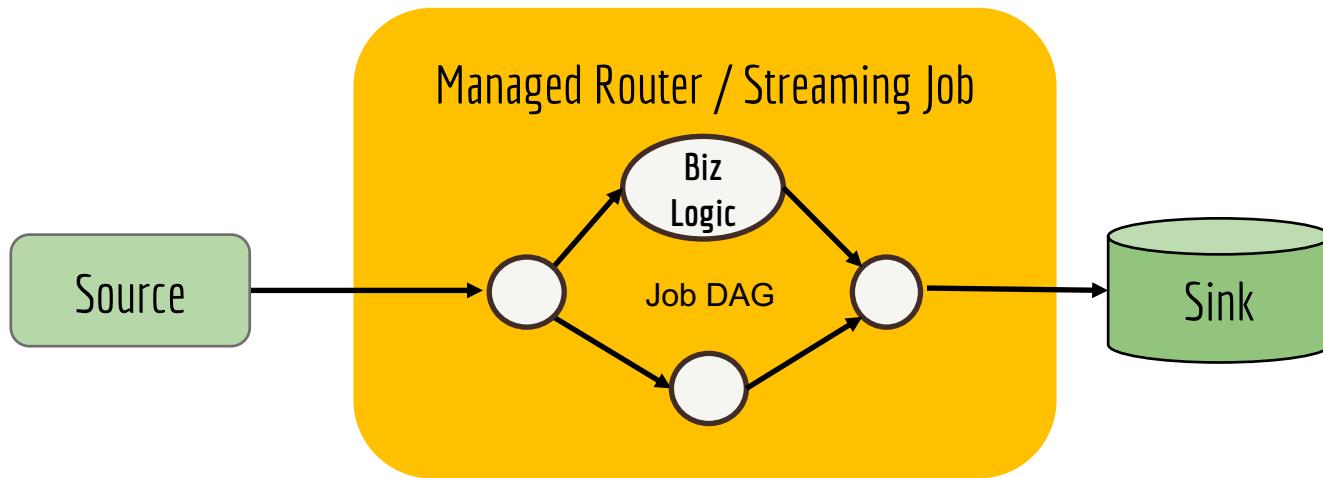
```
ui_A_Clicks_KafkaSource  
.map(transformer)  
.flatMap(outputFlatMap)  
.map(outputConverter)  
.addSink(kafkaSinkA_Topic2)
```

No User Code

2. Script UDF* Component [Static / Dynamic]

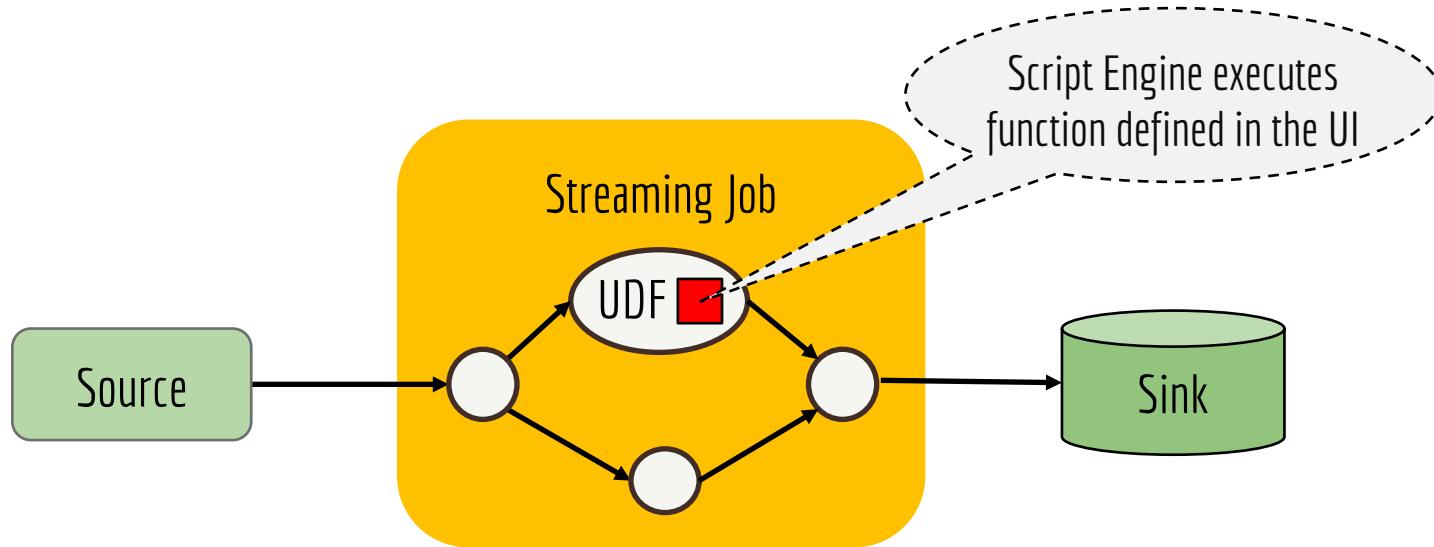
*UDF - User Defined Function

2. Use Case / Motivation - Configurable Business Logic Code for operations like transformations and filtering



2. Pattern: Static or Dynamic Script UDF (stateless) Component

Comes with all the Pros and Cons of scripting engine



2. Code Snippet: Script UDF Component

Contents configurable at runtime

```
val xscript =  
    new DynamicConfig("x.script")  
  
kafkaSource  
    .map(new ScriptFunction(xscript))  
    .filter(new ScriptFunction(xsricpt2))  
    .addSink(new NoopSink())
```

```
// Script Function  
  
val sm = new ScriptEngineManager()  
ScriptEngine se =  
    m.getEngineByName ("nashorn");  
se .eval(script)
```

3. The Enricher

Next 3 Patterns (3-5) Require Explicit Deployment

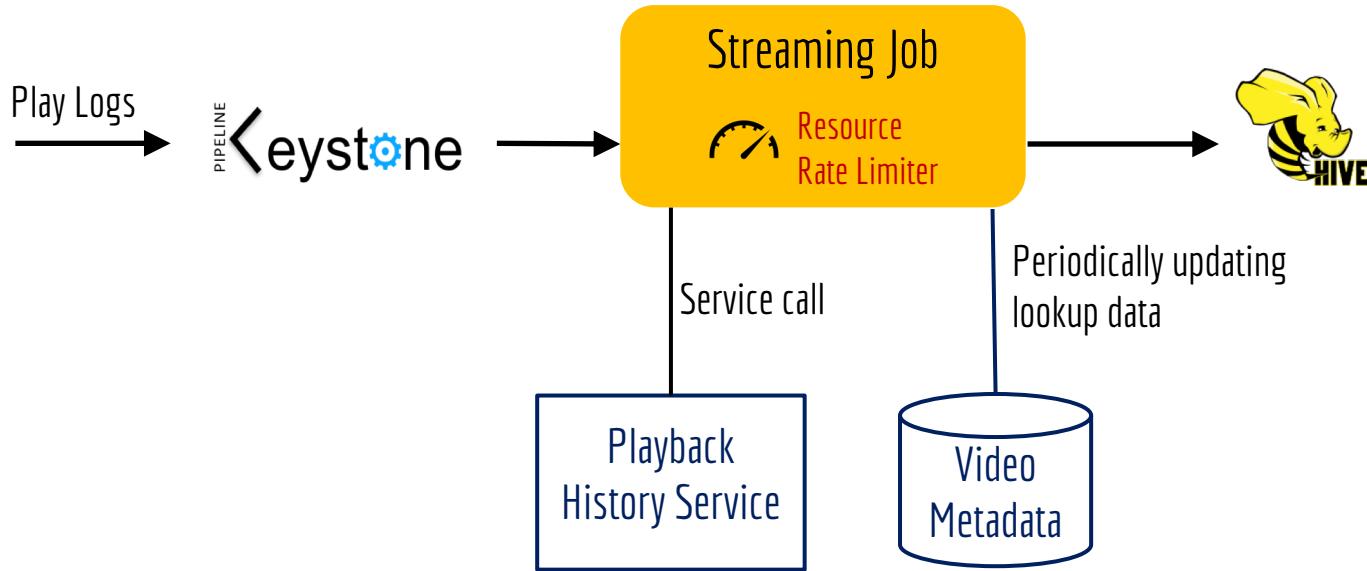
3. User Case - Generating Play Events For Personalization And Show Discovery

The screenshot shows the Netflix homepage with a red background. At the top, there's a navigation bar with the Netflix logo, a search icon, and links for Home, TV Shows, Movies, Recently Added, and My List. To the right are icons for DVD, notifications (with a red dot), and account settings.

Below the navigation bar, there are three sections of recommended shows:

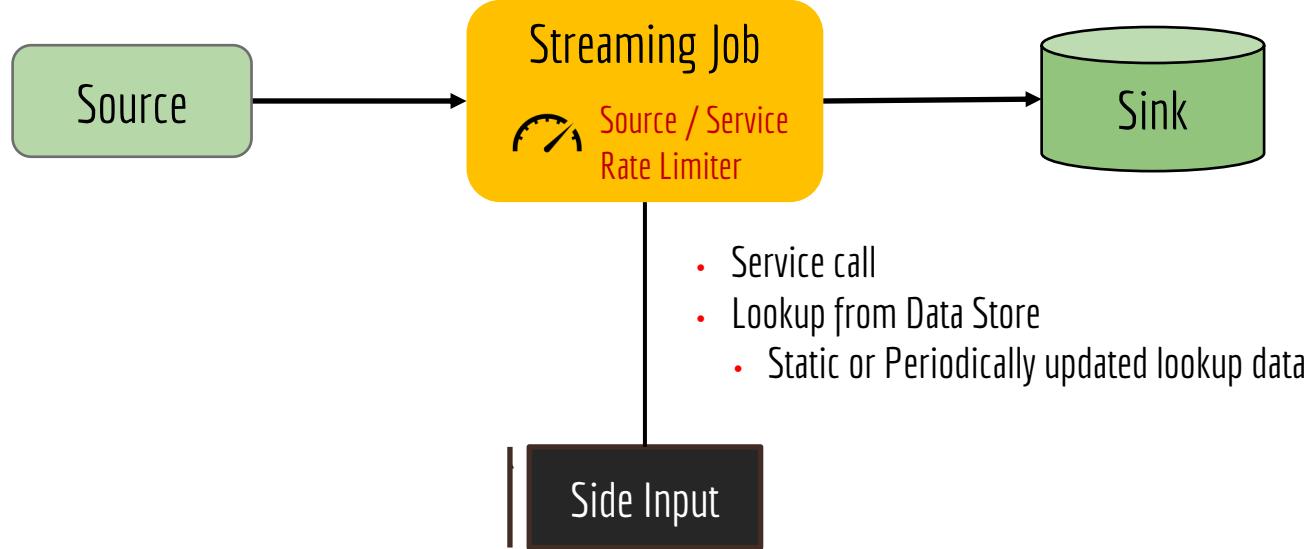
- Acclaimed Writers**: A row of six show thumbnails. From left to right: "The Informant!", "the office", "GLOW", "ORANGE IS THE NEW BLACK", "Master of None", and "OZARK". A large yellow arrow points to the "the office" thumbnail.
- British TV Shows**: A row of six show thumbnails. From left to right: "THE INNOCENTS", "I AM A KILLER", "MILLION POUND MENU", "AMAZING INTERIORS", "JAMES ACASTER REPERTOIRE", and "THE WORLD'S MOST EXTRAORDINARY HOMES".
- Critically-acclaimed TV Shows**: A row of six show thumbnails. From left to right: "COMEDIANS IN CARS GETTING COFFEE", "QUEER EYE", "THE SINNER", "SOMEBODY FEED PHIL", "CHEF'S TABLE", and "The Good Place".

3. Use-case: Create play events with current data from services, and lookup table for analytics. Using lookup table keeps originating events lightweight



3. Pattern: The Enricher

- Rate limit with source or service rate limiter, or with resources
- Pull or push data, Sync / async



3. Code Snippet: The Enricher

```
val kafkaSource = getSourceBuilder.fromKafka("topic1").build()
val parsedMessages = kafkaSource.flatMap(parser).name("parser")

val enrichedSessions = parsedMessages.filter(reflushFilter).name("filter")
    .map(playbackEnrichment).name("service")
    .map(dataLookup)
enrichmentSessions.addSink(sink).name("sink")
```

4. The Co-process Joiner

4. Use Case - Play-Impressions Conversion Rate

The image shows the Netflix homepage with a dark background. At the top, there's a navigation bar with the Netflix logo, a search icon, and links for Home, TV Shows, Movies, Recently Added, and My List. To the right of the search bar are icons for DVD, notifications (9+), and account settings.

Acclaimed Writers

British TV Shows

Critically-acclaimed TV Shows

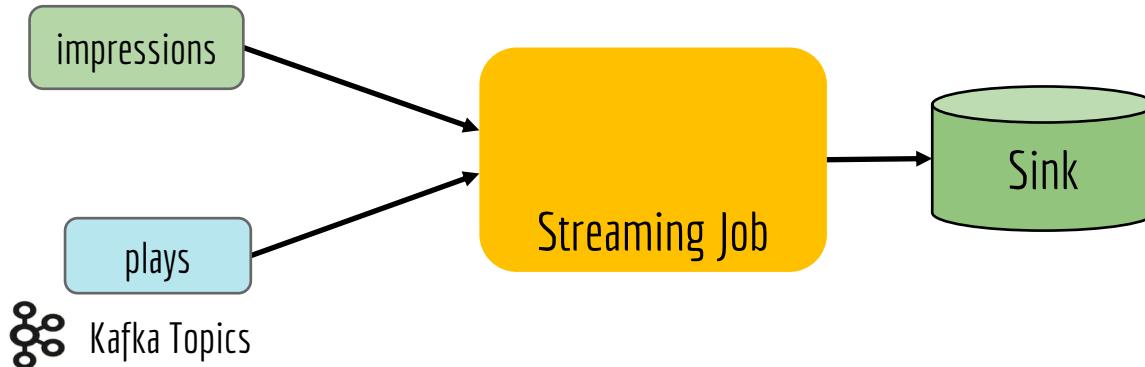
4. Impressions And Plays Scale



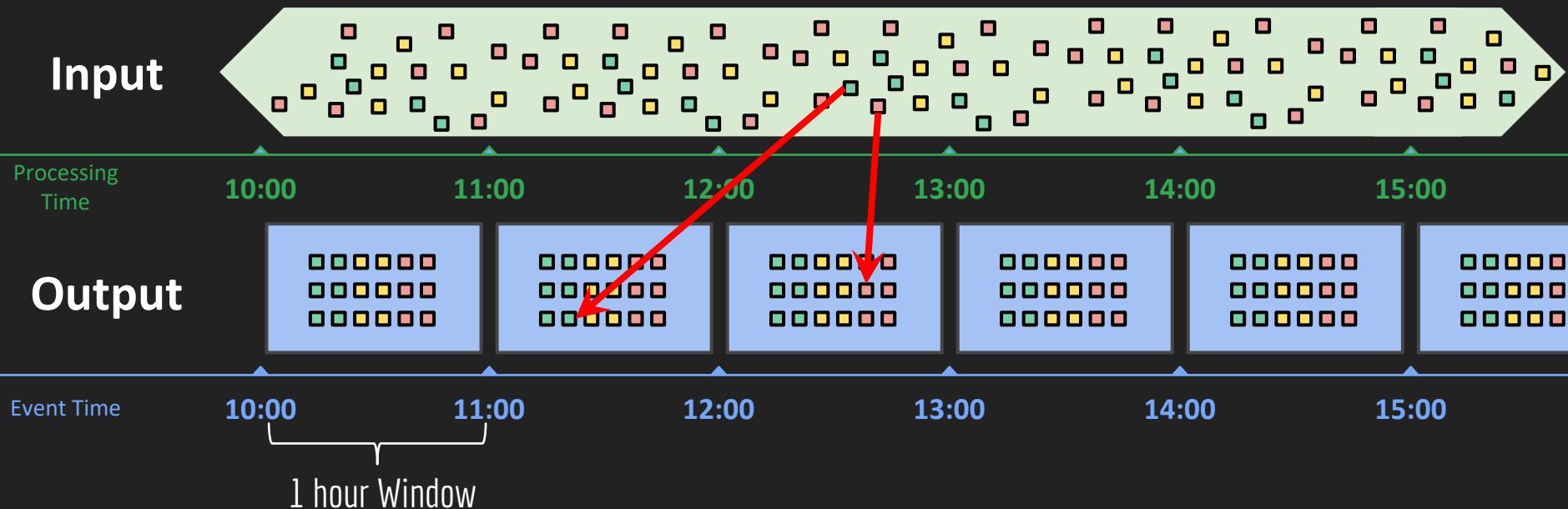
- 130+ **M** members
- 10+ **B** Impressions / day
- 2.5+ **B** Play Events / day
- ~ 2 **TB** Processing State

4. Join Large Streams With Delayed, Out Of Order Events Based on Event Time

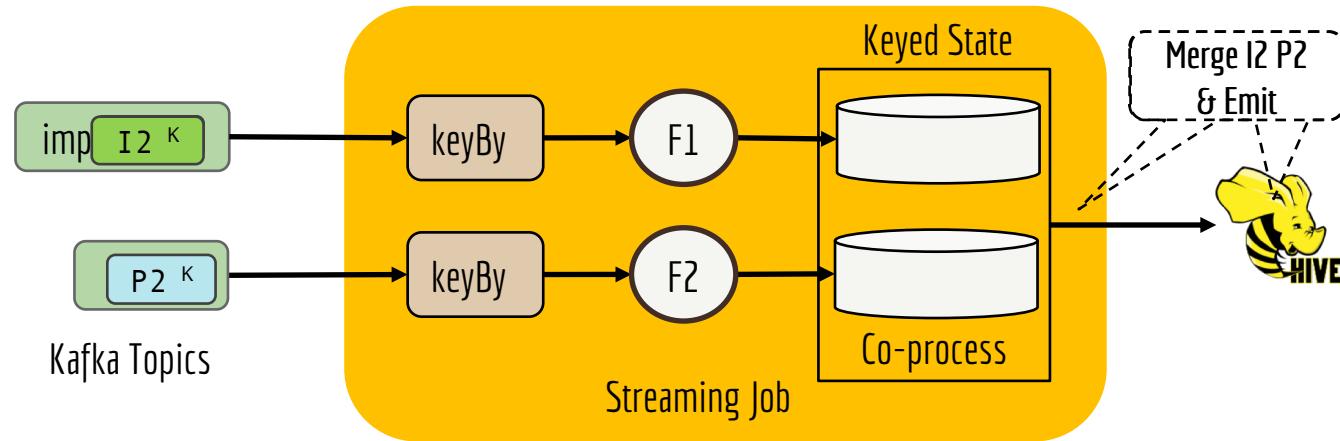
- # Impressions per user play
- Impression attributes leading to the play



Understanding Event Time

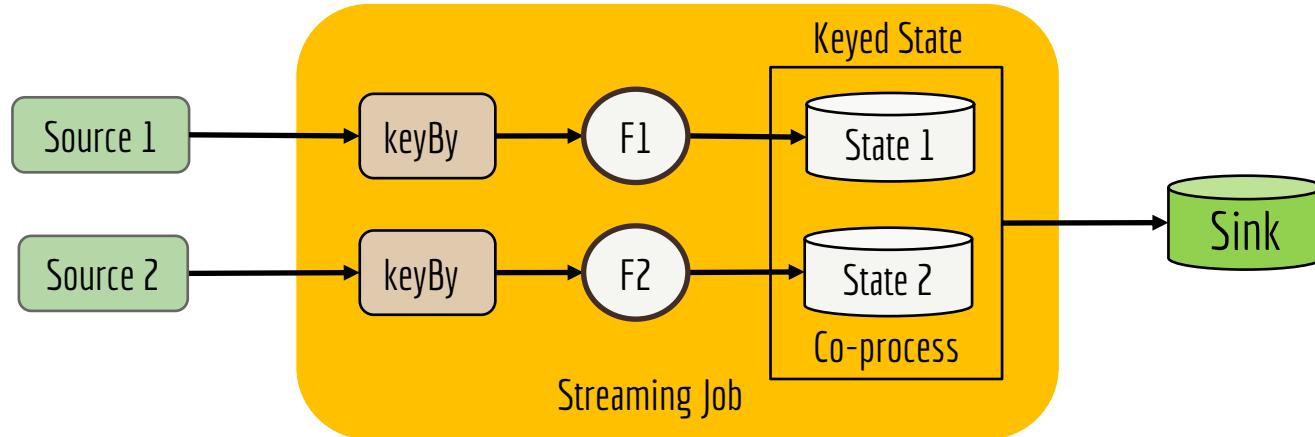


4. Use Case: Join Impressions And Plays Stream On Event Time



4. Pattern: The Co-process Joiner

- Process and Coalesce events for each stream grouped by same key
- Join if there is a match, evict when joined or timed out



4. Code Snippet - The Co-process Joiner, Setup sources

```
env.setStreamTimeCharacteristic(EventTime)
```

```
val impressionSource = kafkaSrc1
    .filter(eventTypeFilter)
    .flatMap(impressionParser)
    .keyBy(in => (s"${profile_id}_${title_id}"))
```

```
val impressionSource = kafkaSrc2
    .flatMap(playbackParser)
    .keyBy(in => (s"${profile_id}_${title_id}"))
```

4. Code Snippet - The Co-process Joiner, Setup sources

```
env.setStreamTimeCharacteristic(EventTime)
```

```
val impressionSource = kafkaSrc1.filter(eventTypeFilter)
    .flatMap(impressionParser)
    .assignTimestampsAndWatermarks(
        new BoundedOutOfOrdernessTimestampExtractor(Time.seconds(10) ) {...})
    .keyBy(in => (s"${profile_id}_${title_id}"))
```

```
val impressionSource = kafkaSrc2.flatMap(playbackParser)
    .assignTimestampsAndWatermarks(
        new BoundedOutOfOrdernessTimestampExtractor(Time.seconds(10) ) {...})
    .keyBy(in => (s"${profile_id}_${title_id}"))
```

4. Code Snippet - The Co-process Joiner, Connect Streams

```
// Connect  
impressionSource.connect(playSessionSource)  
    .process( new CoprocessImpressionsPlays())  
    .addSink(kafkaSink)
```

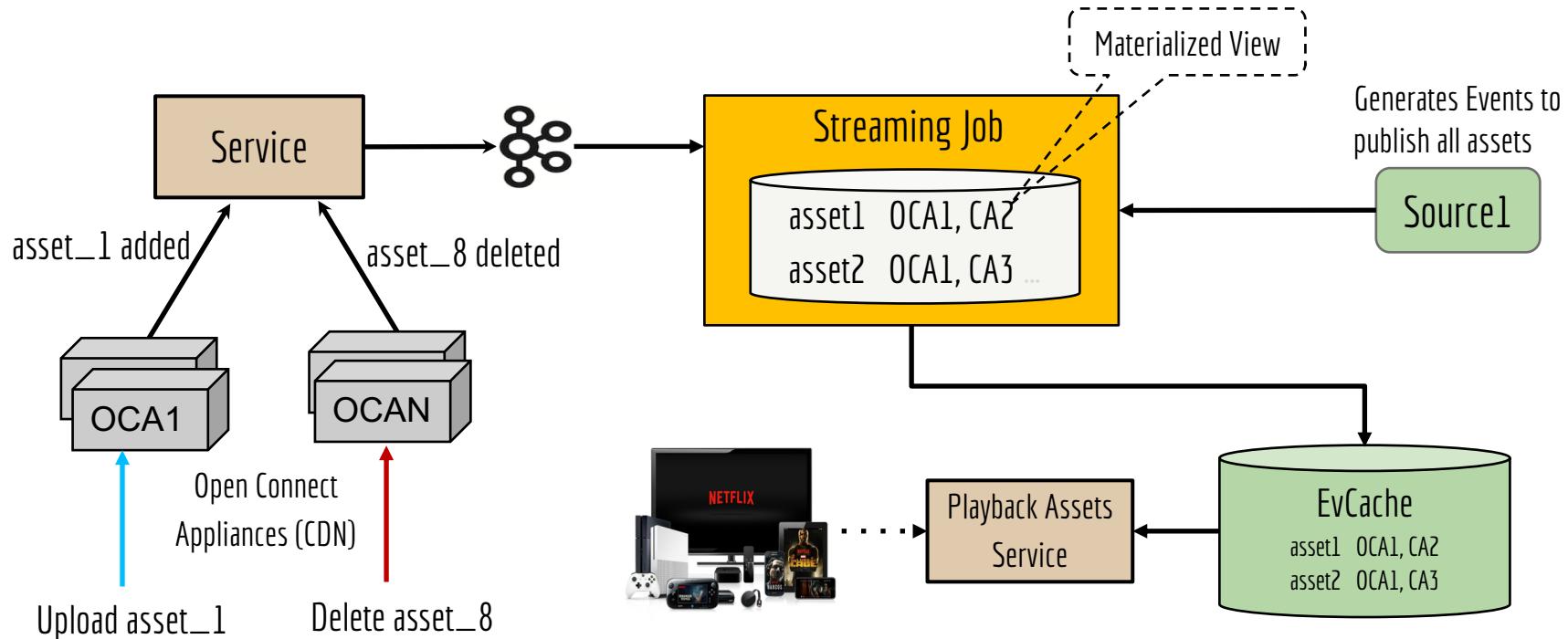
4. Code Snippet - The Co-process Joiner, Co-process Function

```
class CoprocessJoin extends CoProcessFunction {  
    override def processElement1(value, context, collector) {  
        ... // update and reduce state, join with stream 2, set timer  
    }  
    override def processElement2(value, context, collector) {  
        ... // update and reduce state, join with stream 2, set timer  
    }  
    override def onTimer(timestamp, context, collector) {  
        ... // clear up state based on event time  
    }  
}
```

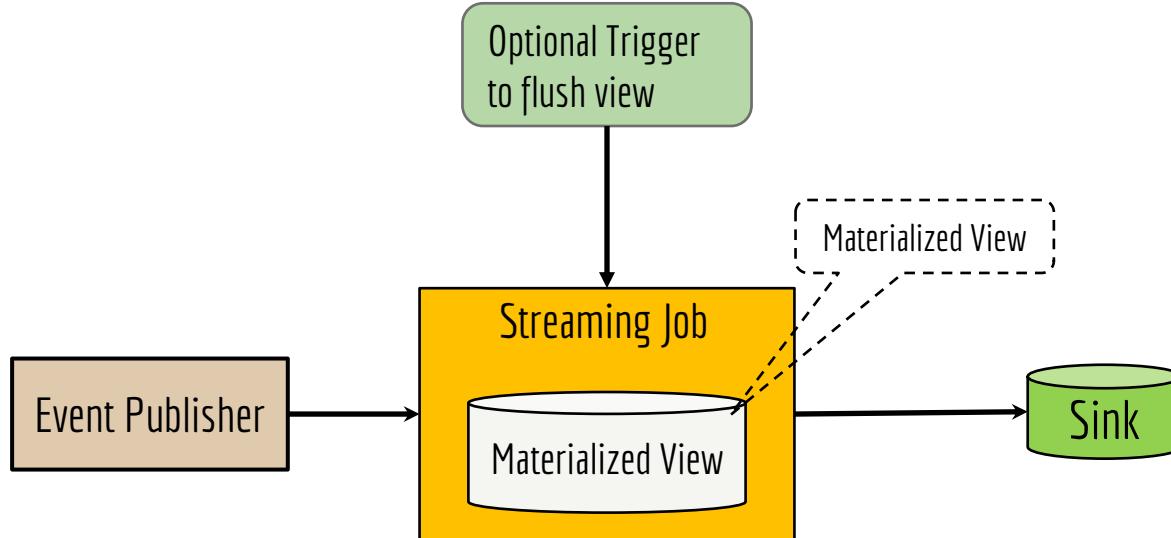
5. Event-Sourced Materialized View

[Event Driven Application]

5. Use-case: Publish movie assets CDN location to Cache, to steer clients to the closest location for playback



5. Use-case: Publish movie assets CDN location to Cache, to steer clients to the closest location for playback



5. Code Snippet - Setting up Sources

```
val fullPublishSource = env.addSource(new FullPublishSourceFunction(),  
TypeInfoParser.parse("Tuple3<String, Integer, com.netflix.AMUpdate>"))  
.setParallelism(1);
```

```
val kafkaSource = getSourceBuilder().fromKafka("am_source")
```

5. Code Snippet - Union Source & Processing

```
val kafkaSource
    .flatMap(new FlatmapFunction())) //split by movie
    .assignTimestampsAndWatermarks(new AssignerWithPunctuatedWatermarks[...]()
    .union(fullPublishSource) // union with full publish source
    .keyBy(0, 1) // (cdn stack, movie)
    .process(new UpdateFunction())) // update in-memory state, output at intervals.
    .keyBy(0, 1) // (cdn stack, movie)
    .process(new PublishToEvCacheFunction())); // publish to evcache
```

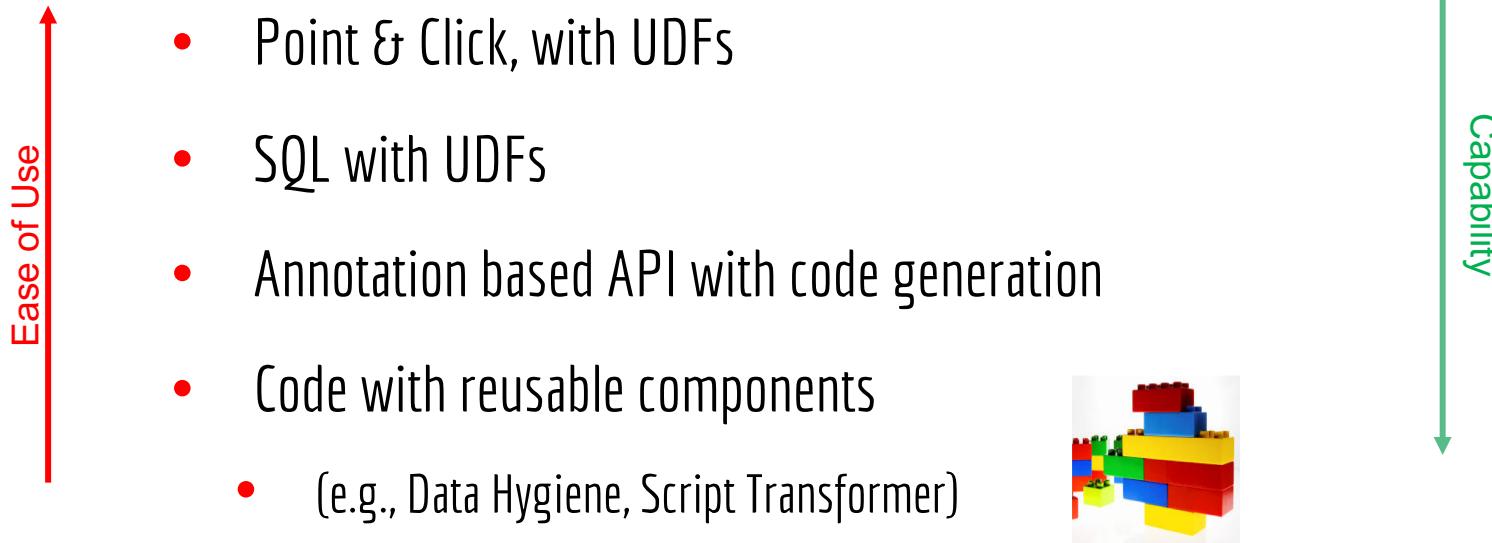
Patterns

| Non-Functional

6. Elastic Dev Interface

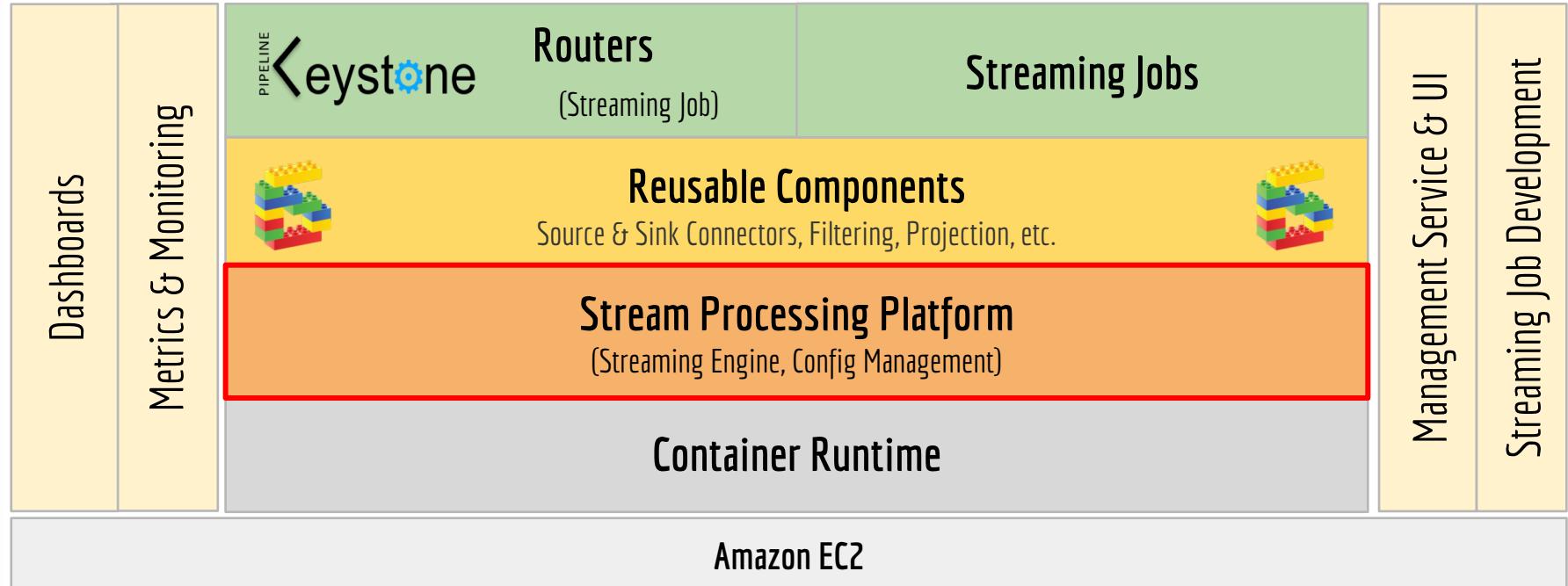
6 Elastic Dev Interface

Spectrum Of Ease, Capability, & Flexibility



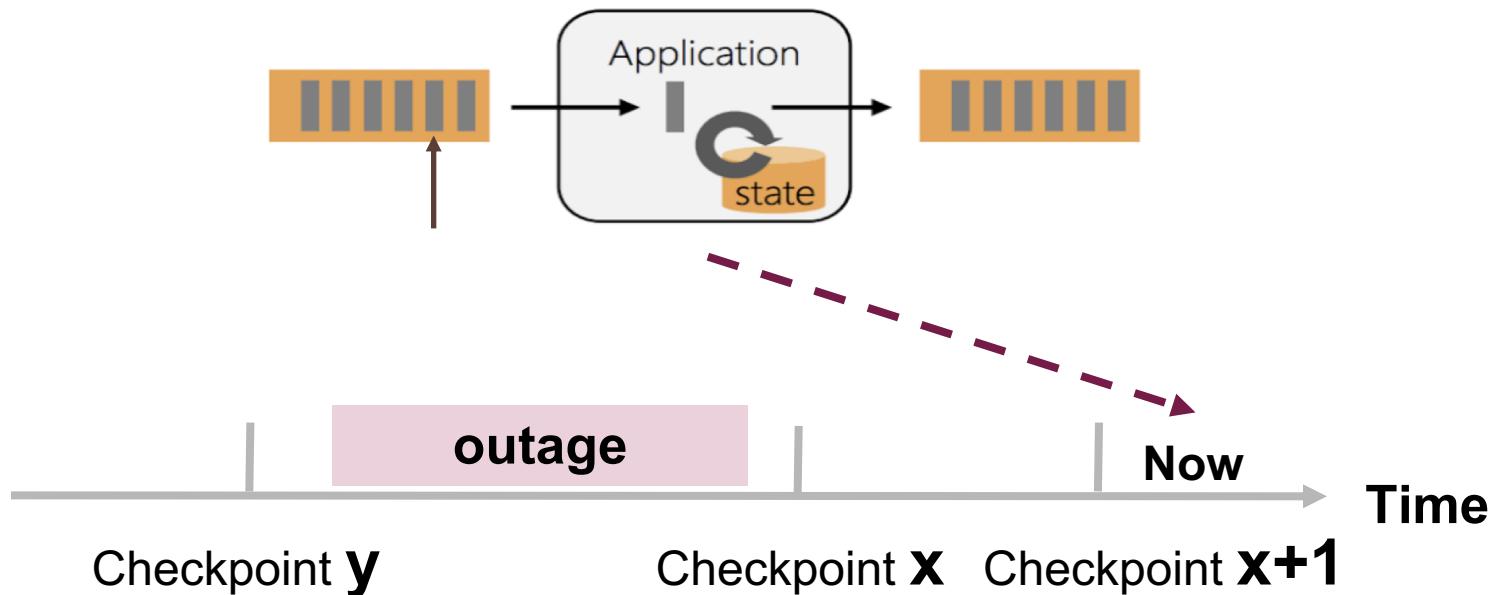
7. Stream Processing Platform

7. Stream Processing Platform (SpaaS - Stream Processing Service as a Service)



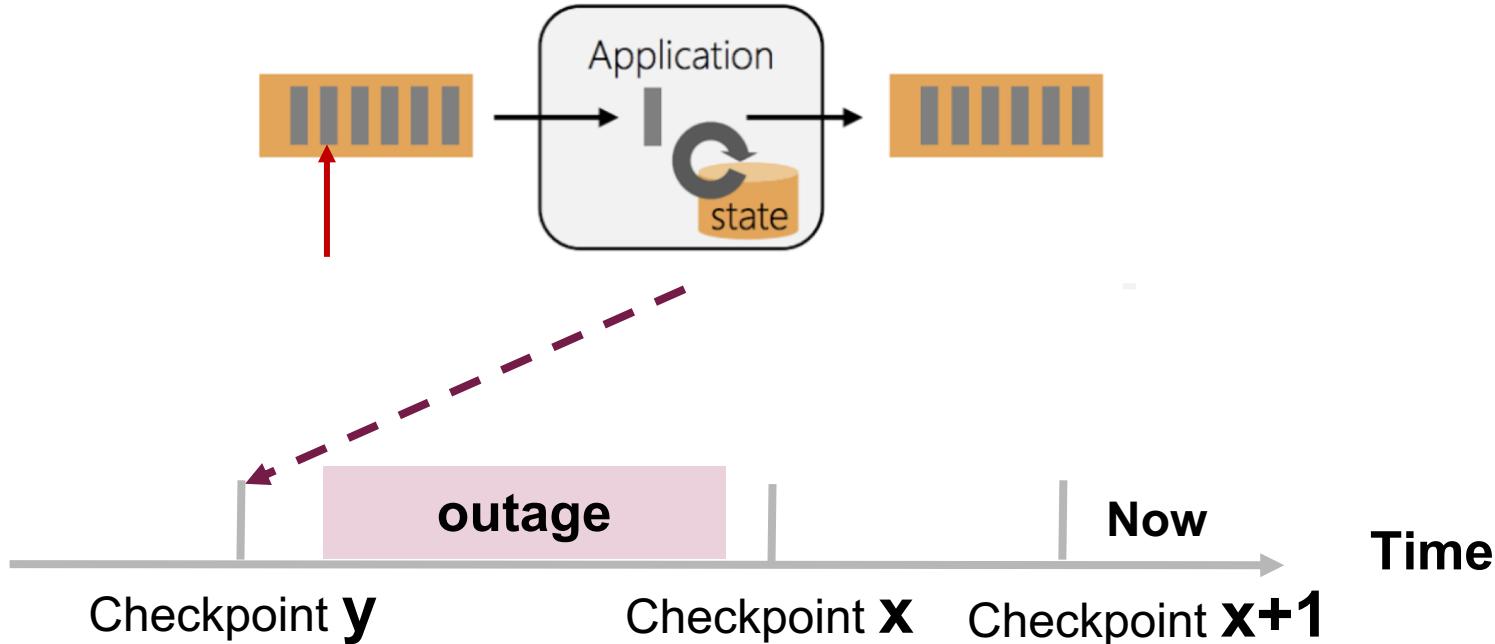
8. Rewind & Restatement

8. Use Case - Restate Results Due To Outage Or Bug In Business Logic



8. Pattern: Rewind And Restatement

Rewind the source and state to a known good state



Summary

Patterns Summary

FUNCTIONAL

1. Configurable Router
2. Script UDF Component
3. The Enricher
4. The Co-process Joiner
5. Event-Sourced Materialized View

NON-FUNCTIONAL

6. Elastic Dev Interface
7. Stream Processing Platform
8. Rewind & Restatement

Thank you

If you would like to discuss more

-  @monaldax
-  linkedin.com/in/monaldax

Additional Stream Processing Material

- Flink at Netflix, Paypal speaker series, 2018- <http://bit.ly/monal-paypal>
- Unbounded Data Processing Systems, Strangeloop, 2016 - <http://bit.ly/monal-sloop>
- AWS Re-Invent 2017 Netflix Keystone SPaaS, 2017 - <http://bit.ly/monal-reInvent>
- Keynote - Stream Processing with Flink, 2017 - <http://bit.ly/monal-ff2017>
- Dataflow Paper - <http://bit.ly/dataflow-paper>