intuit

turbotax quickbooks proconnect mint

# Identity Graph with Flink

Flink Forward SF 2019

Vivek Thakre
Principal Engineer @Intuit

# Intuit

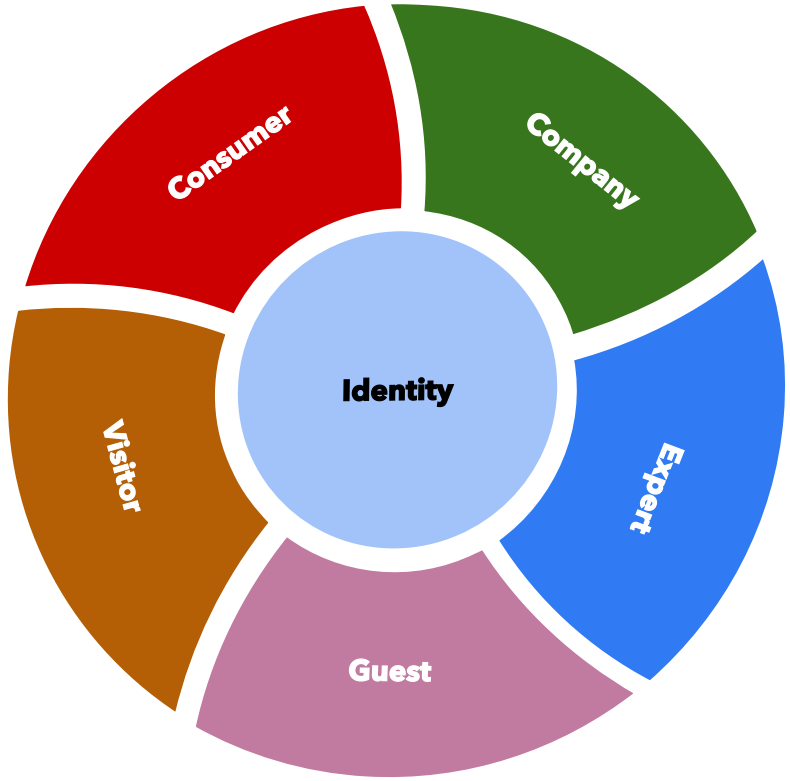
Powering prosperity around the world

# Identity Graph : What are we solving?

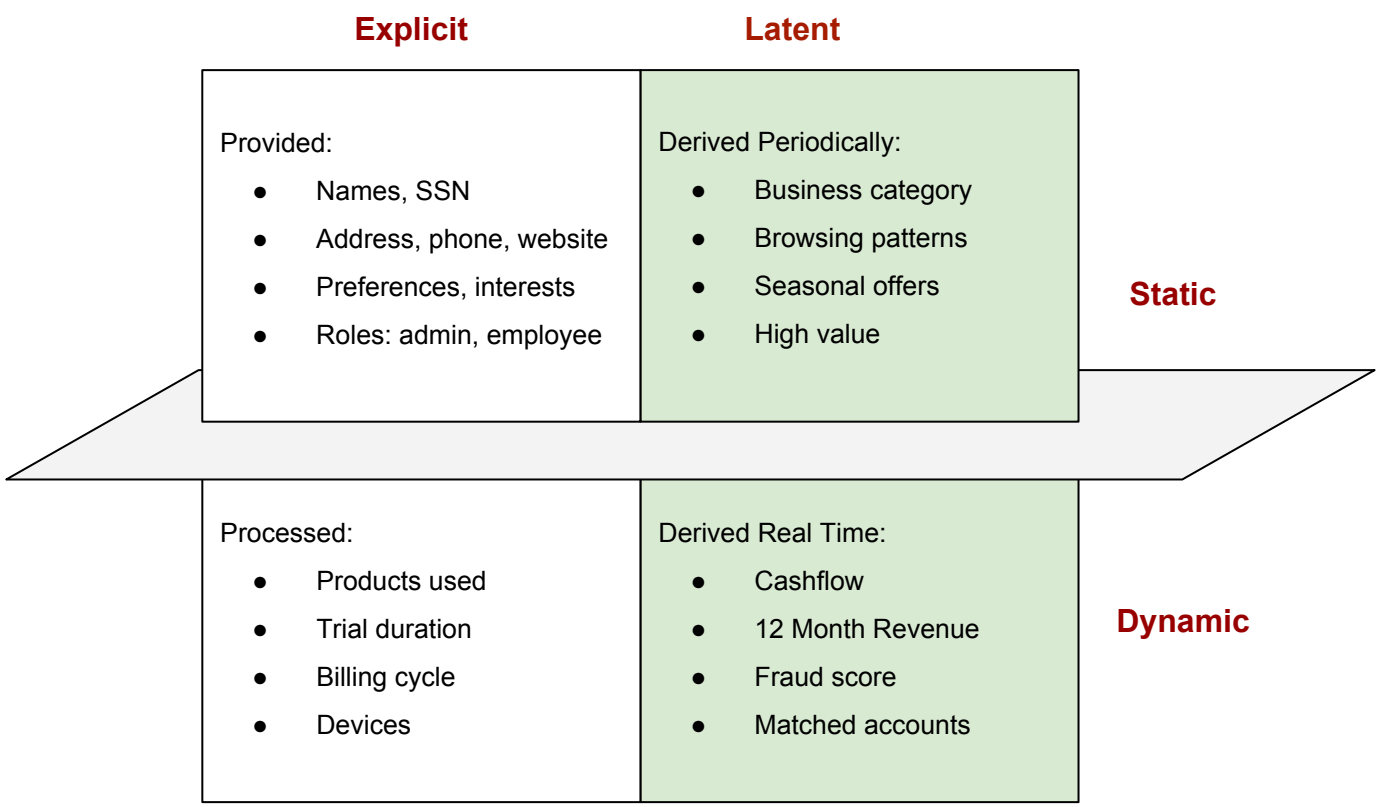
- As an Intuit customer, I keep having to introduce myself to multiple customer care agents
- I get offers for Products and Credit Cards that I already have
- As a Customer Success Agent, I spent minutes collecting information from my customers before I can help them
- As a marketer, I don’t have visibility into my customer’s journey once they become a customer

# Identity Graph

- Goal
  - Personalize the journey for everyone in Intuit's ecosystem
- What
  - Unified Profile Platform and a Service which composes data from various data sources and products into real-time accessible views that personalization services use to change the customer experience in real-time



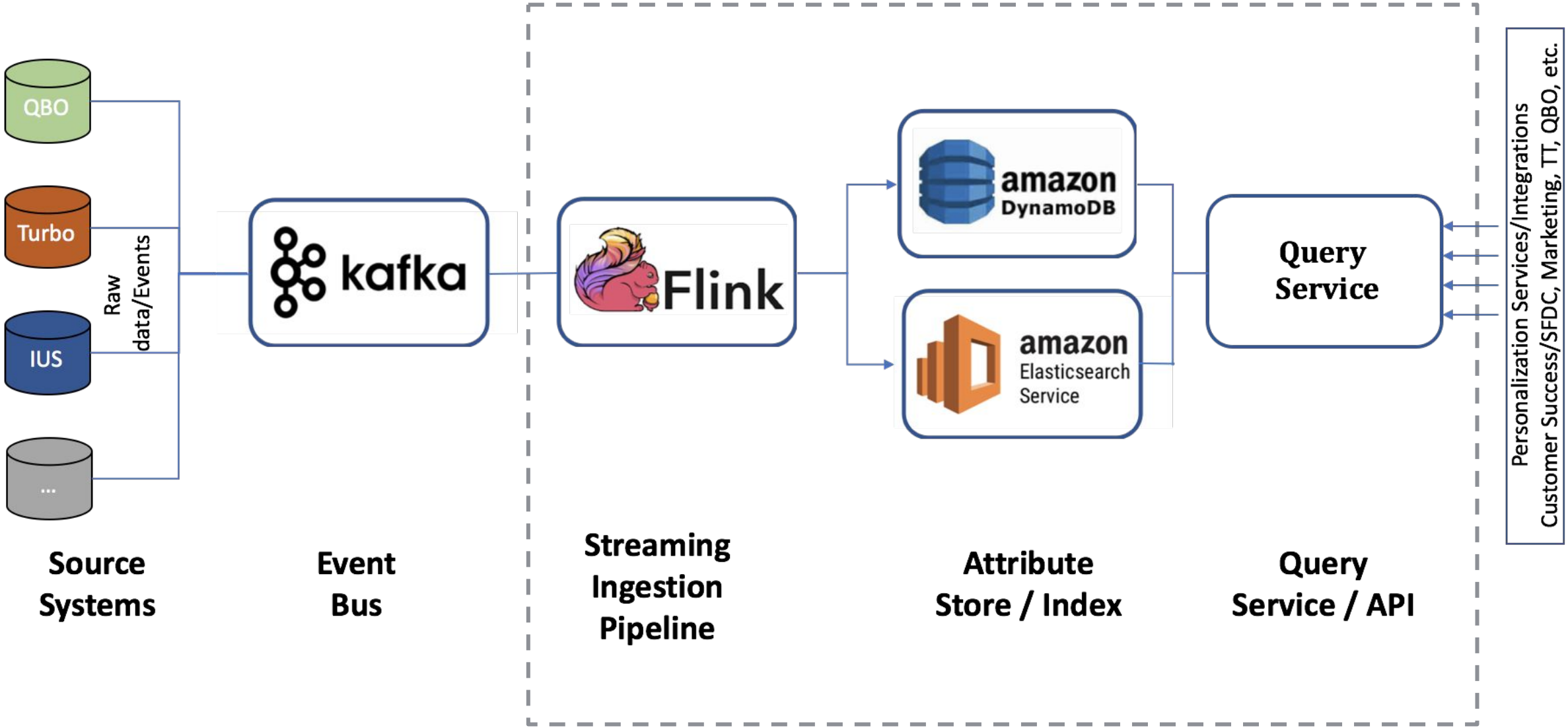
# Profile Attributes and Metadata




## Attribute Metadata

- Name
- Description
- Data Type
- Date Created
- Path
- Source
- Data Classification Level
- Encryption Level
- Identifiable
- Authorization Level
- …

# Identity Graph : High Level Data Flow

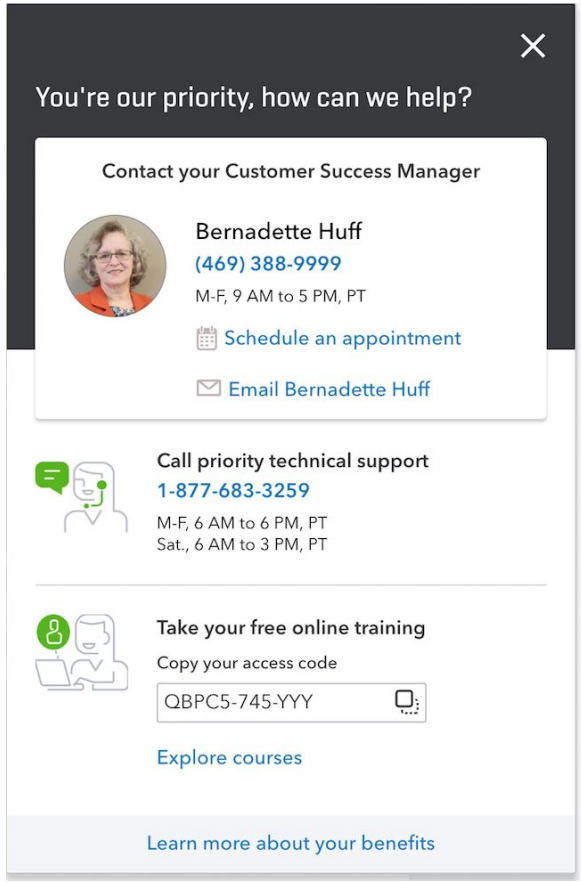


# Use Cases and Ingestion Pipeline

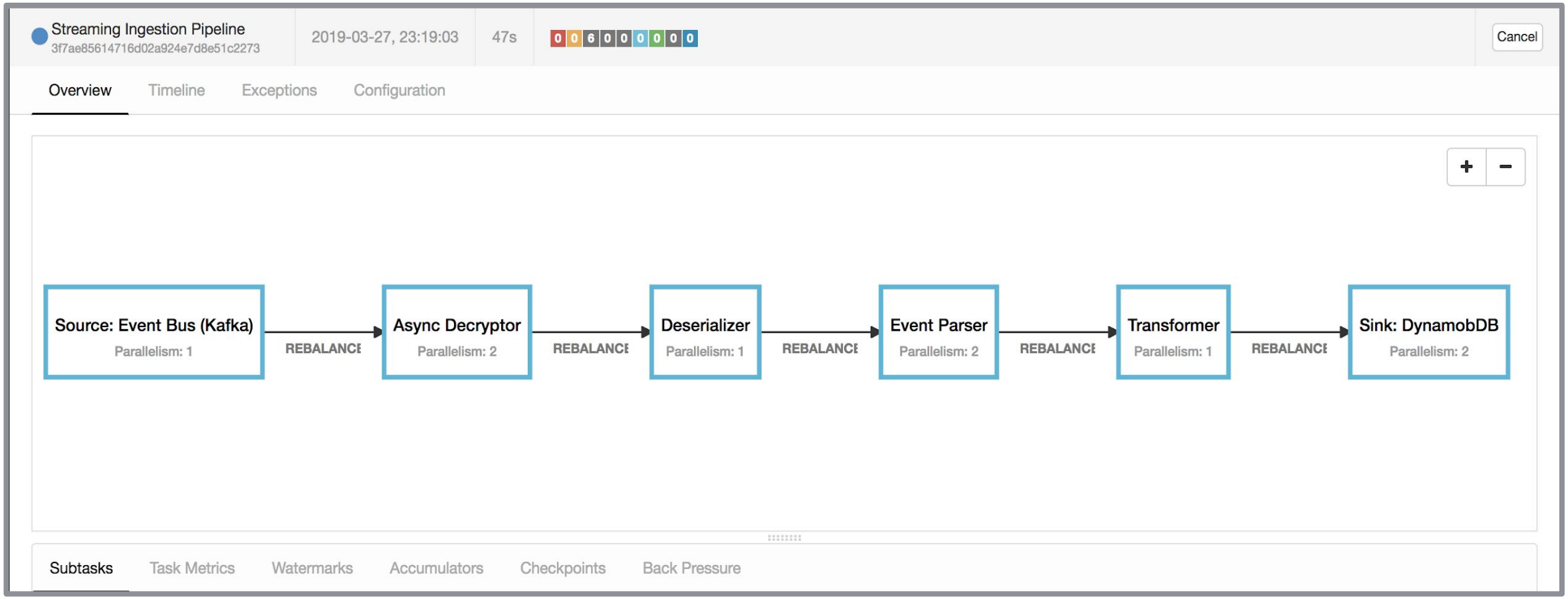
- Different use cases have different requirements
  - Snapshots vs Delta
  - PII vs Non PII Data
  - Point Queries vs Search Capabilities
  - Predefined Schema vs Dynamic
  - Different data reconciliation criteria
- Abstracted common components (Sources, Sinks, Operators)
- Common Components made configurable

# Use Case : Priority Circle

Enable Quickbooks Advanced to display Customer Success Manager details for eligible customers



# Streaming Ingestion Pipeline



# Decryptor using Async IO API

- Higher Streaming Throughput
- Decryptor service supports Async requests
- Implement AsyncFunction with callback
- Configurable failure handler on timeout
- Capacity : backpressure is triggered once the capacity is exhausted
- Stream order needs to be preserved

```
DataStream<String> decryptedInputStream = AsyncDataStream
        .orderedWait(inputStream, new AsyncDecryptor(parameters),
                parameters.getInt("decryptor.timeout", 1000), TimeUnit.MILLISECONDS,
                parameters.getInt("decryptor.capacity", 100)).name("Async Decryptor")
```

# DynamodbSink : Merge Example - Events with same ID

Event 1

```
{
  "identity" : {
    "realmID" : "1000040055"
  },
  "ordinal" : 1553000000,
  "payroll" : {
    "insights" : {
      "daysSinceSignup" : 10,
      "hasContractors" : "true",
      "employee" : {
        "hourlyEECount" : 35
      }
    }
  }
}
```

+

Event 2

```
{
  "identity" : {
    "realmID" : "1000040055"
  },
  "ordinal" : 1553500000,
  "payroll" : {
    "insights" : {
      "daysSinceSignup" : 20,
      "hasContractors" : "true"

    }
  }
}
```

=

Merged Event

```
{
  "identity" : {
    "realmID" : "1000040055"
  },
  "ordinal" : 1553500000,
  "payroll" : {
    "insights" : {
      "daysSinceSignup" : 20,
      "hasContractors" : "true",
      "employee" : {
        "hourlyEECount" : 35
      }
    }
  }
}
```

# Dynamodb Sink with Low Latency

- invoke for each event
- read item
- apply merge function (incoming and existing) - configurable
- update with optimistic locking
  - conditional update with version number
  - on failure retry above steps
- configurable Failure Handlers
- retries with exponential backoff

# Dynamodb Sink with High Throughput

- writes are buffered and merged* for same ids
- during flush
  - batch read for all Ids, apply merge with existing and batch write
- implements checkpointed
- buffer is flushed when
  - size is >=BATCH_SIZE
  - during checkpoint if buffer is not empty
- configurable Failure Handlers
- retries with exponential backoff for unprocessed items

# Domain Event Transformations

- Events are published from different sources with varying schema and format
- Need to transform events into a unified schema
- Transformation rules are domain driven
- Utilizing Jolt library to perform transformations
- Transformations based on EventType and transformation-specs
- Transformation specs in classpath

Jolt : JSON to JSON transformation library written in Java where the "specification" for the transform is itself a JSON document.

https://github.com/bazaarvoice/jolt

# Jolt Transformation

**Jolt Spec** JSON Validate

```
1
2 {
3   "operation": "default",
4   "spec": {
5     "identity_business_realmid": ""
6   }
7  },
8 {
9   "operation": "shift",
10  "spec": {
11    "identity_business_realmid": "identity.realmID",
12    "sb_business_payroll_insights_dayssincepayrollsignup
13    "payroll.insights.daysSinceSignup",
14    "sb_business_payroll_insights_hascontractors":
15    "payroll.insights.hasContractors",
16    "sb_business_payroll_insights_numhourlyemployees":
17    "payroll.insights.employee.hourlyEECount",
18    "sb_business_payroll_payrollusage_qbobillingchannel"
19    "billing.payroll.qboChannel"
20  }
21 },
22 {
23  "operation": "modify-overwrite-beta",
24  "spec": {
25    "identity": {
26      "realmID": "=toString"
27    }
28  }
29 }
```

**Json Input** JSON Validate

```
1 {
2   "identity_business_realmid": 1000040055,
3   "sb_business_payroll_insights_dayssincepayrollsignup":
4   "sb_business_payroll_insights_numhourlyemployees": 0,
5   "sb_business_payroll_insights_hascontractors": 0,
6   "sb_business_payroll_insights_hasworkerscompensation":
7   "sb_business_payroll_payrollusage_qbobillingchannel":
8 }
9
```

**Output / Errors** Transform ☐ **Sort Output?**

```
1 {
2   "identity" : {
3     "realmID" : "1000040055"
4   },
5   "payroll" : {
6     "insights" : {
7       "daysSinceSignup" : 0,
8       "hasContractors" : 0,
9       "employee" : {
10        "hourlyEECount" : 0
11      }
12    }
13  },
14  "billing" : {
15    "payroll" : {
16      "qboChannel" : "other"
17    }
18  }
19 }
```

# Dynamic Transformations : Metadata Service

Profile Self Serve

Search Attributes

Manage Requests

**Admin**

Manage Schema

Manage Mapping Specs

Manage Projected Views

Generate Jolt Specs

Manage Users

Manage Sandboxes

Batch Operations

**Name**

sample_jolt_mappings_specs

**Description**

This is sample specs for demo

**Specification**

```
  },
  {
    "operation": "shift",
    "spec": {
      "identity_business_realmid": "identity.realmID",
      "sb_business_payroll_insights_dayssincepayrollsignup":
      "payroll.insights.daysSinceSignup",
      "sb_business_payroll_insights_hascontractors":
      "payroll.insights.hasContractors",
```

**Specification Type**

JSON

Delete... Cancel Save

# Dynamic Transformations : Event Driven

- Metadata Service Source
  - Rest based Source implen
  - Configurable with Poll Interval
  - Creates Event with Jolt Mappings Specs

```
DataStream<MappingSpecification> joltMappingsSpecsStream = env
                .addSource(new MetadataServiceSource())
```

```
MapStateDescriptor<Void, MappingSpecification> joltMappingsSpecsStateDesc =
new MapStateDescriptor<>("JoltMappinsSpecsDesc",  Types.VOID,
TypeInformation.of(new TypeHint<MappingSpecification>() {}));
```

- Broadcast
  - Describe State for Jolt Mappings Specs
  - Broadcast Source Stream with described state

```
BroadcastStream<Map<String, MappingSpecification>>
broadcastedJoltMappingsSpecsStream = joltMappingsSpecsStream
                .broadcast(joltMappingsSpecsStateDesc);
```

# Dynamic Transformations : JoltTransformationFunction

```
DataStream<ProfileEntity> transformedStream =
inputStream.connect(broadcastedJoltMappingsSpecsStream)
                                .process(new JoltTransformerFunction());
```

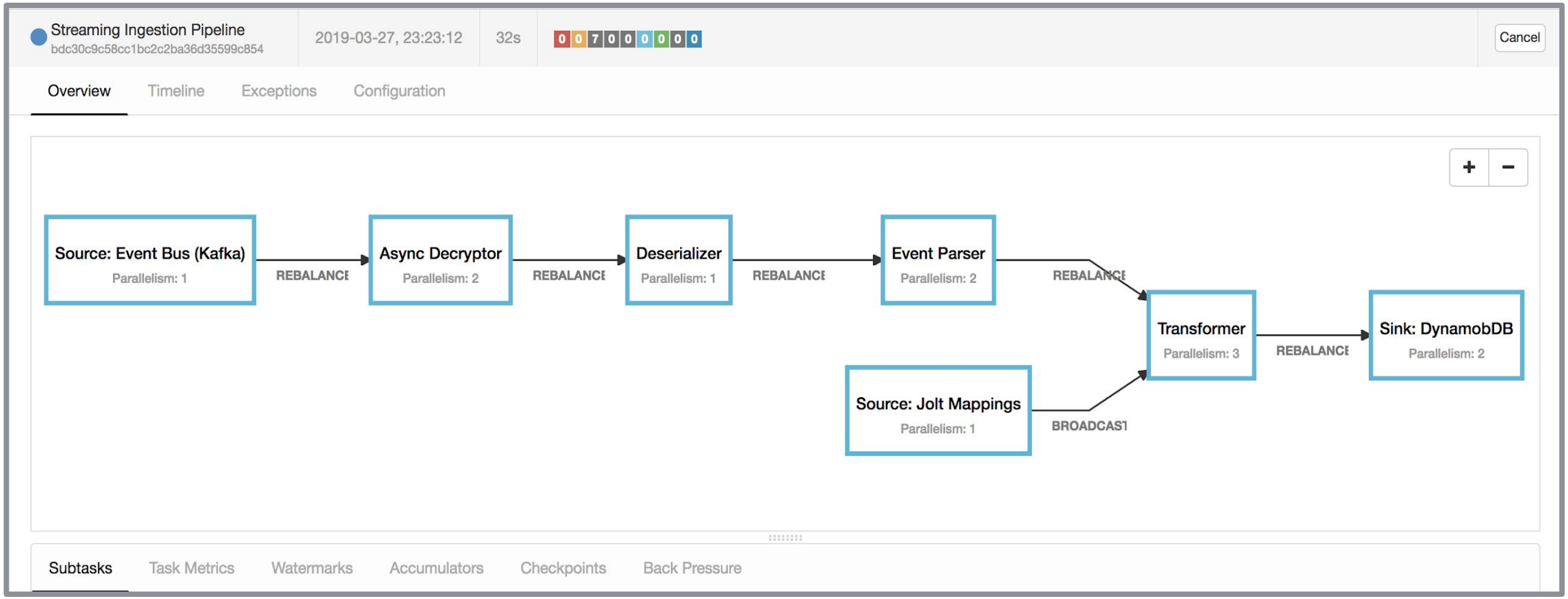
- Connect Broadcasted Stream with Event Stream
- JoltTransformationFunction Implements BroadcastProcessFunction
- BroadcastProcessFunction : processBroadcastElement
  - update broadcasted state

```
BroadcastState<Void, MappingSpecification> bcJoltMappindsSpecsState =
ctx.getBroadcastState(joltMappingsSpecsStateDesc);
bcJoltMappindsSpecsState.put(null, joltMappingsSpecs);
```

- BroadcastProcessFunction : processElement
  - processElement : read state and apply transformation on event

```
MappingSpecification joltMappingsSpecs =
ctx.getBroadcastState(joltMappingsSpecsStateDesc)
                                .get(null);
```

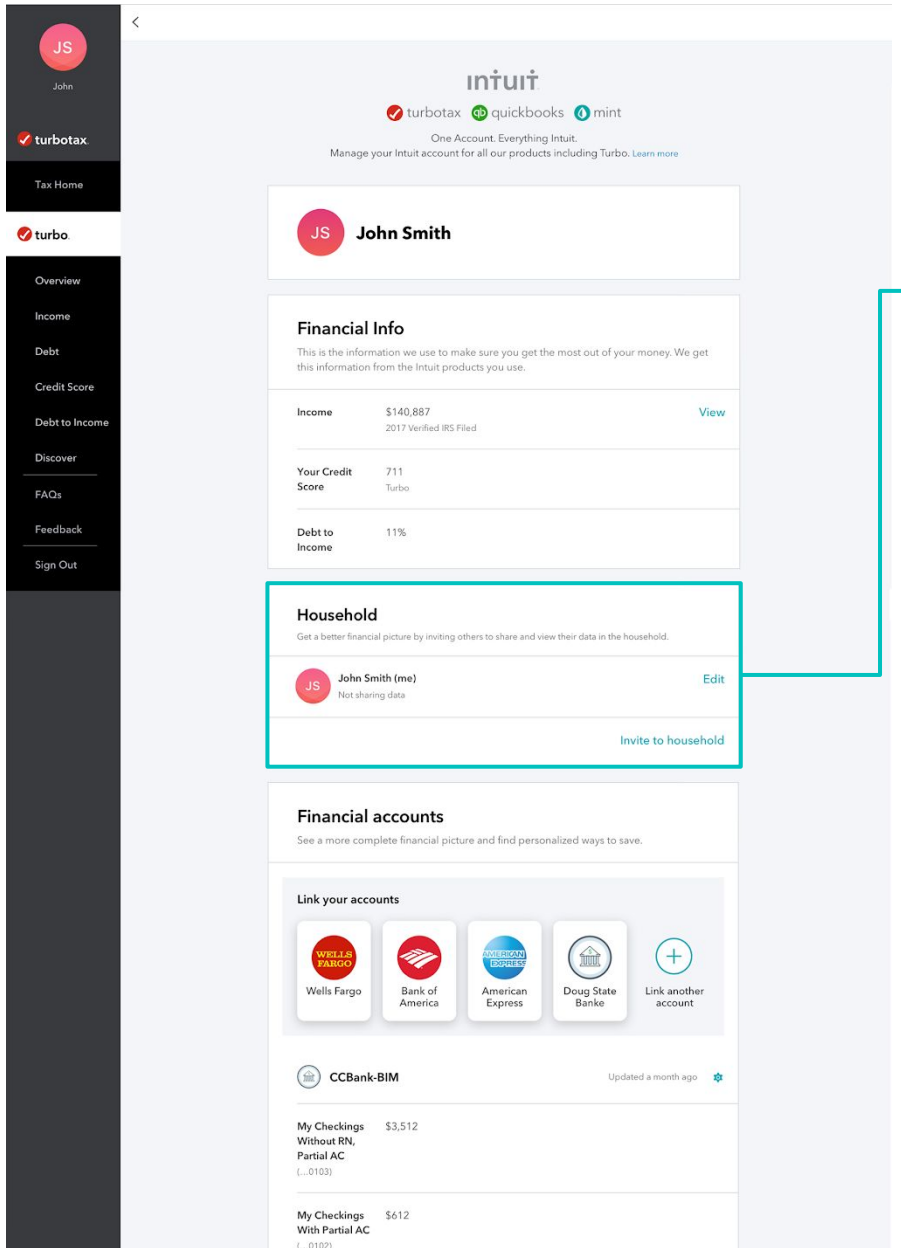
# Ingestion Pipeline with Dynamic Transformation



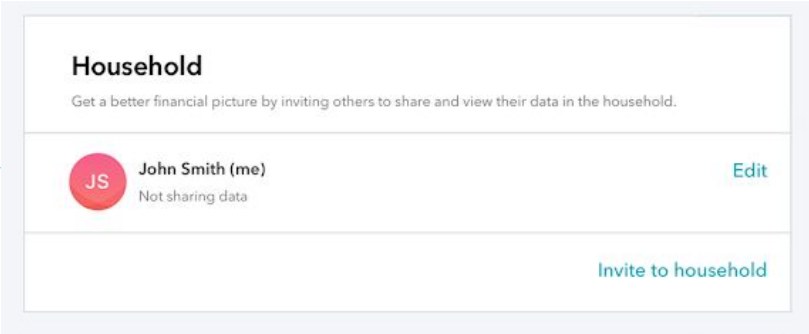
# Dynamic Transformation with Broadcast

- ‘Side Input’ is better fit for this pattern
- For static and slowly evolving data
- FLIP-17 Side Inputs for DataStream API

# Use Case : Turbo Household




## HOUSEHOLD WIDGET




## Purpose

**Allow users to link their profiles in order to share financial data and see a household view of their finances.**

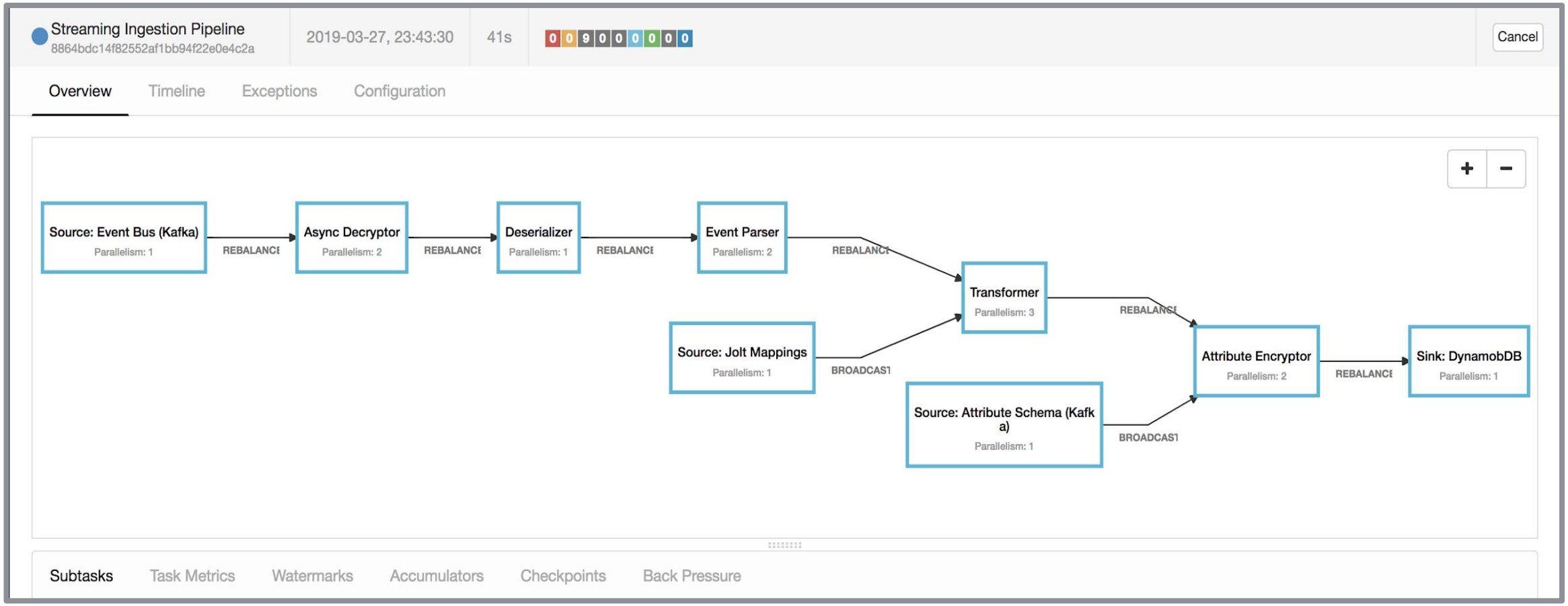
**Release 1.0**

Base capability to allow individuals to connect their profiles to share their **credit scores** via *one way* data sharing.

# Attribute Level Encryption

- Attribute schema updates along with metadata is published to event-bus
- Broadcast metadata to Encryptor Function
- Based on metadata, determine highly sensitive attributes and get encryption keys
- Encryption is performed utilizing external service

# Ingestion Pipeline with Attribute Level Encryption



# Use Case : VCI or Contact Info

## Search Results

Guided Tour | Help for this Page

Search Feeds

**Records**

Reports (0)
People (0)
Accounts (5)
Leads (0)
Documents (0)
Contacts (4)
Attachments (0)

**Search All**

united | Search Again | Options...

### Accounts (5)

| Action | Account Name | Account Site | Phone | Account Owner Alias |
|---|---|---|---|---|
| Edit | Express Logistics and Transport | | (503) 421-7800 | YName |
| Edit | United Oil & Gas, UK | | +44 191 4956203 | YName |
| Edit | University of Arizona | | (520) 773-9050 | YName |
| Edit | United Oil & Gas Corp. | | (212) 842-5500 | YName |
| Edit | United Oil & Gas, Singapore | | (650) 450-8810 | YName |

### Contacts (4)

| Action | Name | Account Name | Account Site | Phone | Email | Contact Owner Alias |
|---|---|---|---|---|---|---|
| Edit | Mr. Josh Davis | Express Logistics and Transport | | (503) 421-7800 | j.davis@expressl&t.net | YName |
| Edit | Ms. Jane Grey | University of Arizona | | (520) 773-9050 | jane_gray@uoa.edu | YName |
| Edit | Ms. Ashley James | United Oil & Gas, UK | | +44 191 4956203 | ajames@uog.com | YName |
| Edit | Ms. Babara Levy | Express Logistics and Transport | | (503) 421-7800 | b.levy@expressl&t.net | YName |

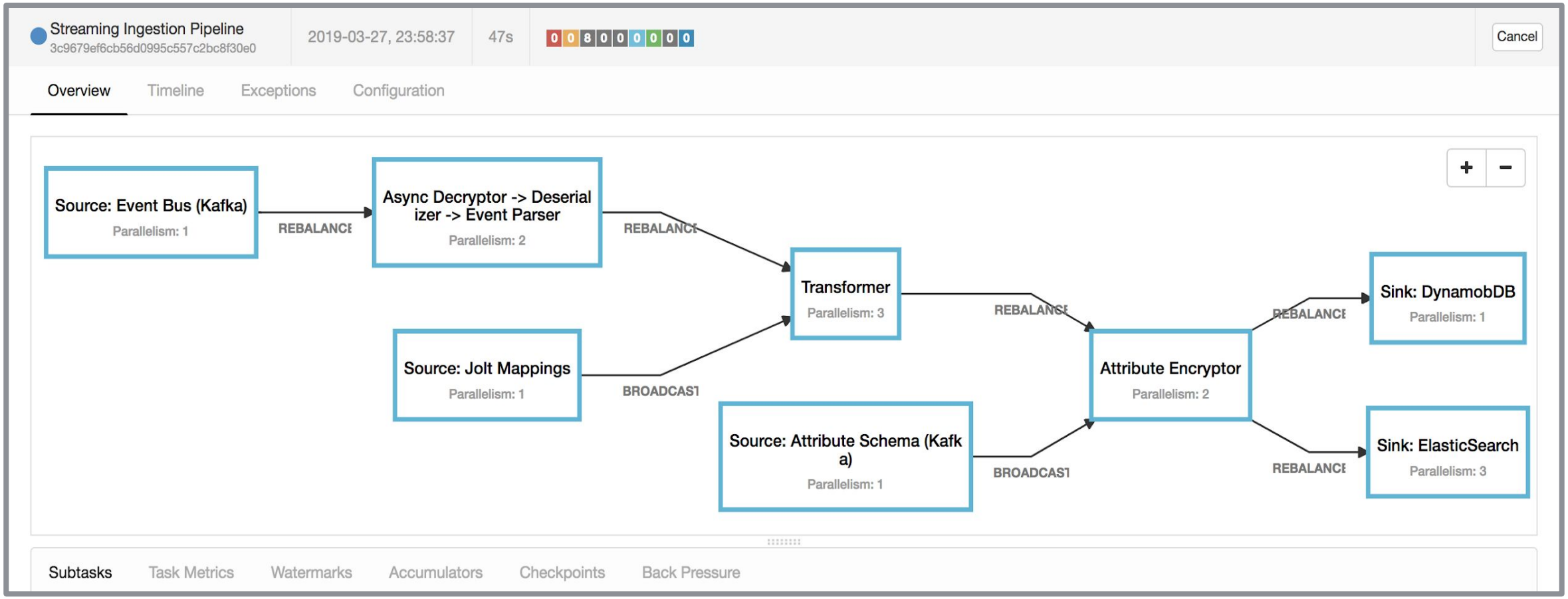
Search All

# ElasticSearch Sink

- Utilize side-output streams
- Elasticsearch Connector that is bundled with Flink
- Implements checkpointed
- IndexableAttributes : documentId, attribute and value

```
OutputTag<IndexableAttributes> indexableAttribute =
        new OutputTag<IndexableAttributes>("indexableAttribute"){};
DataStream<IndexableAttributes> attributeIndexerStream =
        encryptedStream.getSideOutput(indexableAttribute);
attributeIndexerStream.addSink(new ElasticSearchSink(parameters));
```

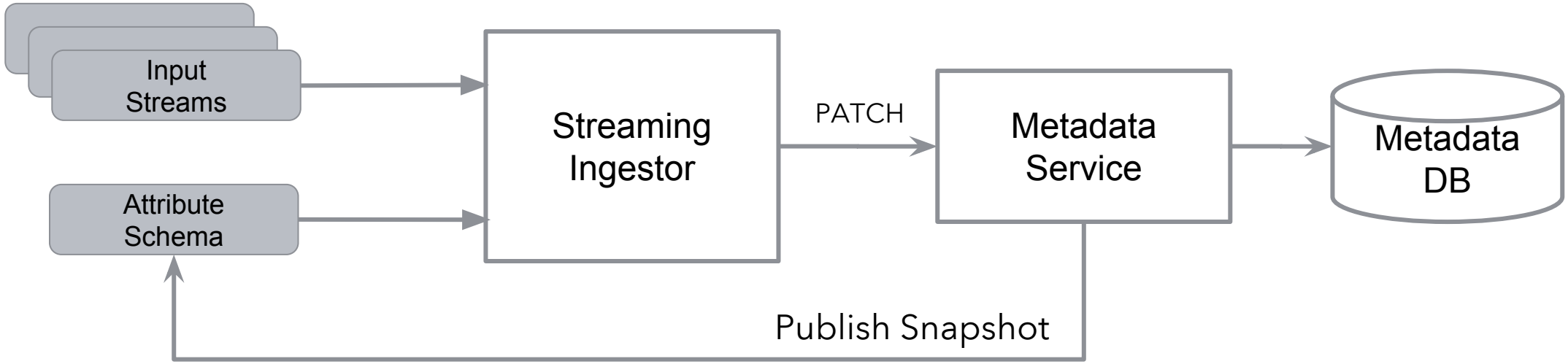
# Ingestion Pipeline with Search Capabilities - Side Output



# Use Case : Rapid Experimentation

- Experimentation Platform run 1000s of experiments
- Data Scientists and analyst run models and create and derive new attributes
- Data is ingested with new attributes but its not a part of schema
- Schema needs to be updated dynamically
- New attributes needs to be discoverable and usable within 2s

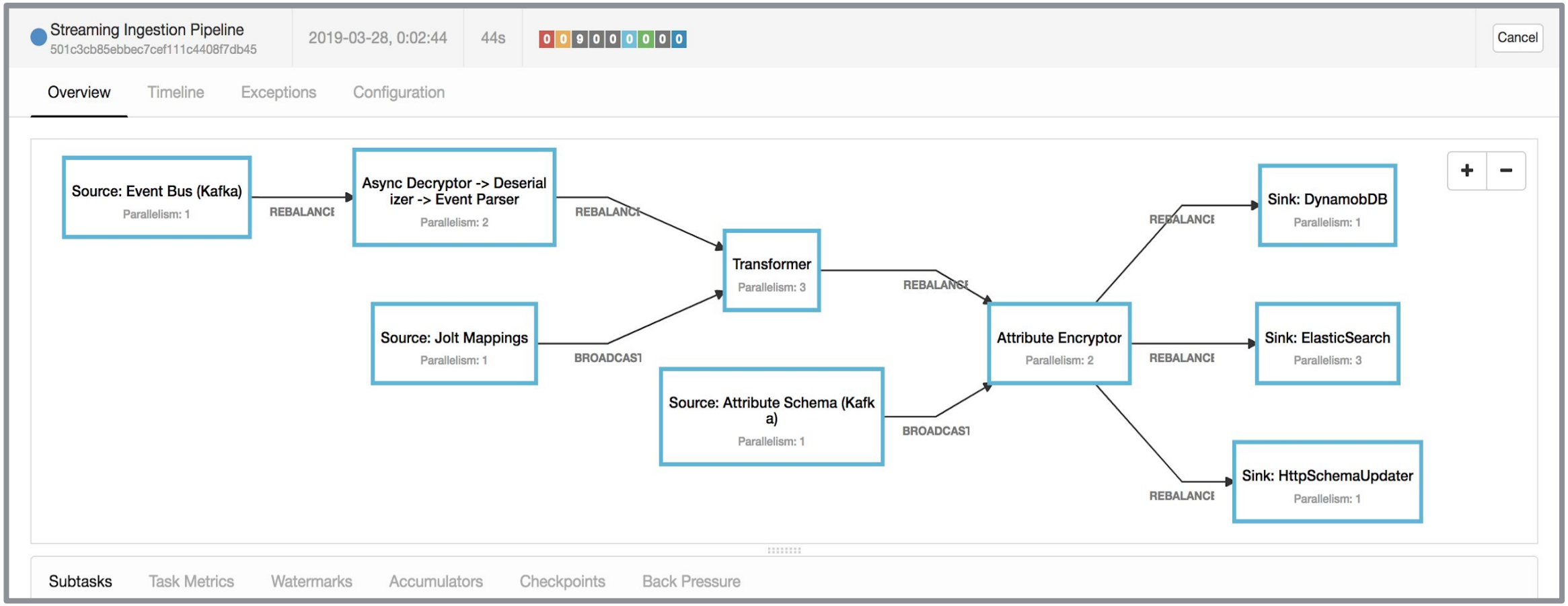
# Dynamic Schema Update



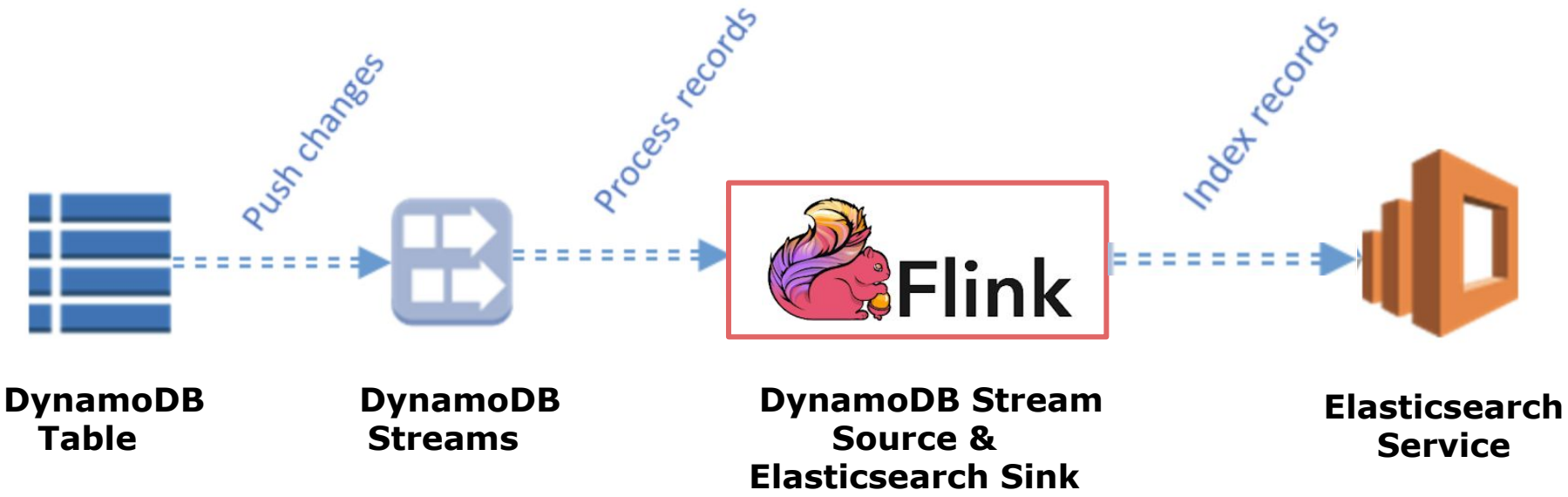
# Dynamic Schema Update : HttpSchemaUpdaterSink

- Compute diff between existing schema and incoming event schema
- Utilize Side-output stream to create schema change stream
- Update metadata via http patch call to Metadata Service
- Created HttpSchemaUpdaterSink
  - Buffer and eliminate duplicate schema updates
  - Implements ProcessingTimeCallback
    - register future time with new update
  - Flush requests when
    - updates count >= MAX_UPDATES or
    - triggered by timer after SCHEMA_UPDATE_INTERVAL

# Streaming Ingestion Pipeline



# Dynamodb Streams to Elasticsearch




- Utilizing FlinkDynamoDBStreamsConsumer in Flink 1.8
- FLINK-4582

# Streaming Ingestion Pipeline

Building blocks of Pipeline, Abstracted as Components, New Components are created with requirements

## Sources

- Kafka Consumer
- Jolt Specs Mappings
- Dynamodb Stream Consumer
- Attribute Schema

## Operators

- Deserializer
- Decryptor
- Transformer
- Event Parser
- Encryptor

## Sinks

- Persistor (Dynamodb)
- Attribute Indexer (ElasticSearch)
- Schema Updater (HTTP)

# Q&A

Thank You