OBJECT PARTNERS

Data Management with Avro, Schema Registry and Apache Atlas*

Exceptional results with Exceptional people

Why, A Story

"Pthhh, just publish some JSON"

Why, Really

Know your data

- source and definition

Know where it's been and who it's seen

- tranformation and enrichment

Know what parts are PII/PHI

- security and auditing

Be a good neighbor

- downstream integration

Avro

Schemas are good

- well defined types, extensible

Prepackaged with evolution in mind

- compability checks: forwards, backwards, full

Automatic compatibility resolution

Language bindings for all

- java, .Net, python, nodejs, go ... more

Avro Best Practices: Defaults

ALWAYS set a default, all fields optional

```
{
"name" : "age",
"type" : "int",
"doc" : "The number of years a user has lived",
"default" : "-1"
}
```

Optional using null

```
{
"name" : "age",
"type" : ["null", "int"],
"doc" : "The number of years a user has lived",
"default" : "null"
}
```

Avro Best Practices: enums & aliases

NEVER an enum

- compatibility breaks with a new symbol

Aliases are not implemented across all clients

Avro Best Practices: Doc

ALWAYS add docs to your types/fields

- Make it meaningful
- Can be parsed to documentation (e.g. avrodoc tool)

Avro Best Practices: Build Lint

Validate your constraints at build time.

- build fails if any contraints do
- perform compatibility checks

Avro Best Practices: Reuse

Reuse common datatypes

- Versioned artifacts with schema archives
- Use simple models, avoid deep nesting

Avro Best Practices: Registry

Finally, use a schema registry

Avro FYI

Avro JSON output can be unusual when unions are involved (null defaults).

```
{"age": {"int": "32"}}
```

Kafka with Avro

Use the schema registry

One schema per topic

Always backwards compatible

Breaking compatibility change: expand ➥ collapse

Avro for the key too

Kafka Connect: Avro to S3/HDFS

Avro works great in data lakes, good compatibility

Can create large files with many messages

- save $$$

Topic schema included, albeit modified

Works on "stream" time

- low volume topics may have significant consistency delays

Apache Atlas, A Data Governance Tool

Fairly new, version 0.8.1

Essentially maintained by Hortonworks

Uses Titan for graph, moving to JanusGraph

Kafka for notifications

HBase, Zookeeper and Elasticsearch also required.

Apache Atlas, The What

Tracks and records data lineage

Create data taxonomies

Tag and categorize data (PII, etc.)

UI and Rest API included

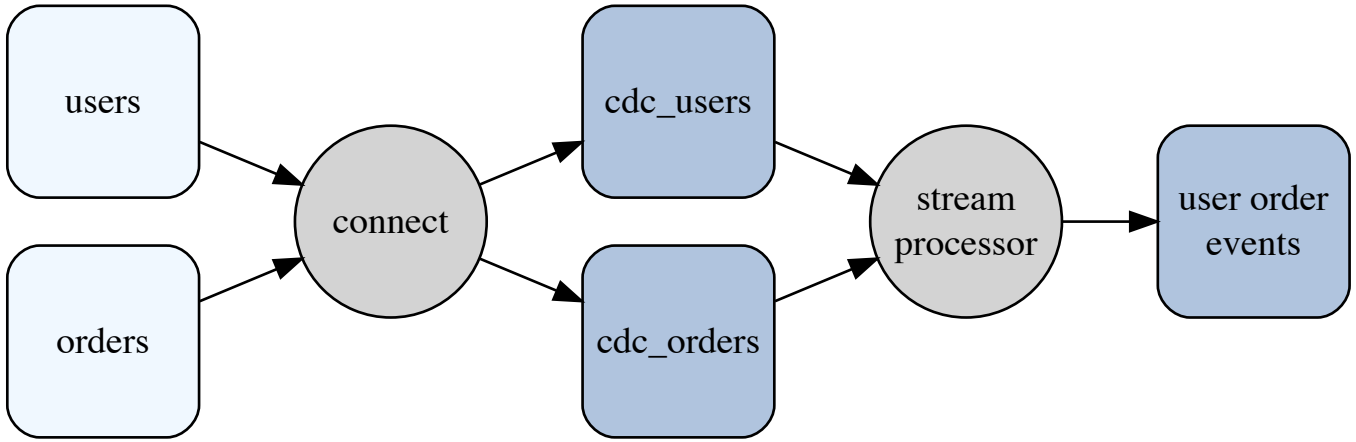
Hooks into other systems for metadata collection

- Hive, HBase, Storm, etc.

Does even more...

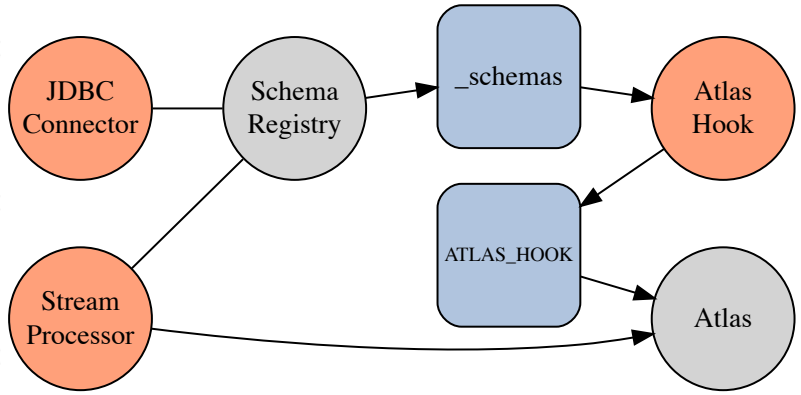
Kafka, Avro and Atlas

Goal: I want to show people how the data flows and not be involved with keeping it up to date.



Integrating Kafka POC

- Integration via schema registry
- listen to `_schemas` topic, publish to `ATLAS_HOOK` topic
- Atlas API for the rest



Integrating Kafka POC: Metadata

Create kafka and avro specific metadata model

Example, A Kafka Connect Process

```
{
  "name": "kafka_connect_process",
  "description": "A kafka connector process",
  "superTypes": [ "Process" ],
  "typeVersion": "1.0",
  "attributeDefs": [
  {
      "name": "cluster",
      "typeName": "kafka_cluster",
      "cardinality": "SINGLE",
      "isOptional": false,
      "isIndexable": true,
      "isUnique": false,
      "description": "The kafka cluster where the topic is defined"
    }
  ]
}
```

Integrating Kafka POC: JDBC Connector

Use topic naming convention to capture some metadata

➥ *Look into using simple transform*

[category]-[type]-[dbName]-[table] => `cdc-db-demo-accounts`

Set the connector's `topic.prefix` to include the metadata: `"topic.prefix": "cdc-db-demo-"`

Extract in the hook and use to create Atlas entities:

```
String subject = extractTopicName(schemaValue); //from _schemas topic
List<String> parts = SUBJECT_SPLITTER.splitToList(subject);

String category = parts.get(0);
String type = parts.get(1);
String database  = parts.get(2);
String table = parts.get(3);
```

Integrating Kafka POC: Kafka Streams

Use Atlas API to create the kafka streams process entity.

- Need to lookup topics to make references

Schema registry integration will capture the topics.

Must manually specify input/output topics in stream configuration.

```
stream:
  inputTopics: [topic-in-1, topic-in-2]
  outputTopics: [topic-out]
```

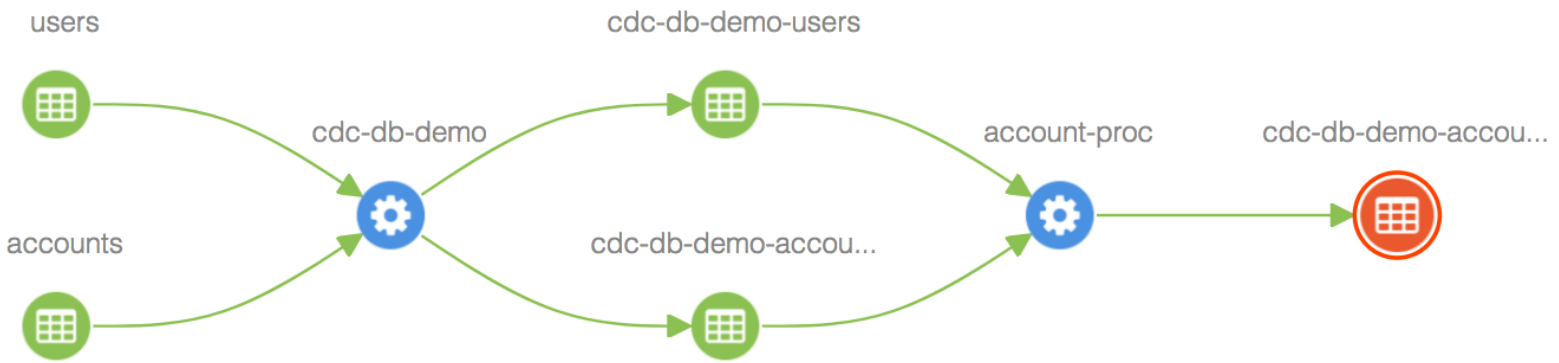
Integrating Kafka POC: KSQL?

KSQL currently does not support avro ☹

Don't worry, support is on the agenda ☺

Perfect fit otherwise, captures the input and output topics naturally

Demo


users
accounts
cdc-db-demo
cdc-db-demo-users
cdc-db-demo-accou...
account-proc
cdc-db-demo-accou...

Questions?