Kafka Connect

https://z.umn.edu/connect_ideaa

Kafka

https://z.umn.edu/connect_ideaa

Kafka

— Data integration platform
— Open Source (Apache)
— Highly scalable, reliable and available

https://z.umn.edu/connect_ideaa

Kafka (cont.)

— Stores data in Topics
  — Retain for as long as you want
  — Can be read multiple times
  — Immutable and data order is guaranteed

https://z.umn.edu/connect_ideaa

Connect

https://z.umn.edu/connect_ideaa

Connect

I want to take data from here and move it there

— How to connect to Kafka
— How to get the data
— Where to put the data
— How to find new data
— How often to check for new data

https://z.umn.edu/connect_ideaa

Connect

— Connect wraps up that use case in to a simple tool

— Boiler plate is handled by Connect, all you provide is a small configuration file

— Connect then spins up a worker that does the work

https://z.umn.edu/connect_ideaa

Workers

Sources and Sinks

https://z.umn.edu/connect_ideaa

Sources

In Connect terminology, a 'Source' is a worker that gets data from somewhere and puts it in Kafka.

- Databases
- Key/Value stores
  - e.g., Redis
- Message queues
  - e.g., SQS
- Files
- More!

https://z.umn.edu/connect_ideaa

Sinks

Workers that take data from Kafka and put it somewhere else are called Sinks.

You can have as many sinks as you want for a single set of data.

- Database
- key/value store
- Message queue
- File
- External system
  - e.g. Splunk
- More!

https://z.umn.edu/connect_ideaa

Actual Usage

Some examples of Connect in current use at UMN

— Source data from a vendor's MSSQL database
— Sink application log data to Splunk
— Sink event stream data to Amazon SQS
— Source Amazon SQS data as a work queue

https://z.umn.edu/connect_ideaa

Connect Worker Options

— Lots!

— We're going to demo JDBC Source and Sink

https://www.confluent.io/hub/

https://z.umn.edu/connect_ideaa

Demo

Move data
From `hoteldev`
Into `hoteltst`

https://z.umn.edu/connect_ideaa

LGD72NJC52

```
ASR_DEV_USER_WHIT0694@//oracle-hotel-dev.oit.umn.edu:1521/hoteldev.oit 03-AUG-22 >
```

Hoteldev

LGD72NJC52

```
ASR_DEV_USER_WHIT0694@//oracle-hotel-tst.oit.umn.edu:1521/hoteltst.oit 03-AUG-22 >
```

Hoteltst

Real World Considerations

— As with most demos, that looked really easy.

— But as you watched it, you probably though of lots of real world complications

— It will help if we understand what our Source and Sink workers are doing.

https://z.umn.edu/connect_ideaa

What Our Source Worker Does

Our Source Query

```
SELECT
  *
FROM
  people
```

What Connect Executes

```
SELECT
  *
FROM
  people
WHERE
  updated_at > last_time_i_checked
  OR
  id > biggest_id_i_have_seen
```

https://z.umn.edu/connect_ideaa

- Any rows returned by the query are written to Kafka topic
- Won't return hard deletes
- `updated_at` needed to find changed records
- `id` needed to find new records

What Our Sink Worker Does

Our Sink Command

```
MERGE INTO PEOPLE USING dual on ( id=21 )
WHEN MATCHED THEN
  UPDATE SET name='updated',
    created_at=1659533399130,
    updated_at=1659533399130,
WHEN NOT MATCHED THEN
  INSERT
    (id, name, created_at, updated_at)
  VALUES
    (21, 'updated', 1659533399130, 1659533399130)
```

— Requires a way to uniquely identify a row

https://z.umn.edu/connect_ideaa

Some possible solutions

— Use `ora_rowscn`
— Do bulk imports
— CDC
— Roll your own

https://z.umn.edu/connect_ideaa

Use `ora_rowscn`

— Oracle identifies transactions with a unique number

— Used by Connect to identify Inserts and Updates

Pros

— Get updates and inserts without primary key or timestamp columns

Cons

— Won't get hard deletes

— Still need a primary key to use `upsert`

— Oracle specific

https://z.umn.edu/connect_ideaa

Bulk imports

— Connect can get all records at once -- "Bulk" import
— Sink does a "Bulk" write of all records to the target

Pros

— Get updates and inserts **and deletes**
— You have recreated PS Snap! 🎊

Cons

— Not a near-live solution
— You have recreated PS Snap! 🎊

https://z.umn.edu/connect_ideaa

CDC

— Uses DB replication logs, not the tables, as the Source

— Sink replays those logs against your targets

Pros

— Get updates and inserts **and deletes**

— Near-live

Cons

— Requires escalated database access

— Config is more complex

https://z.umn.edu/connect_ideaa

Roll Your Own

— Connect was meant to make 80% of common tasks easier

— But for special cases you can bypass Connect and write your own

Pros

— Implements exactly the logic you need

Cons

— Writing and maintenance is on you

https://z.umn.edu/connect_ideaa

Datatypes

What if I want to move data from Oracle to MySQL, which has different datatypes?

- The Kafka Sink worker does a pretty good job of converting between databases
- Or, you could `cast` in your Source query to get things in to a datatype that both database share

https://z.umn.edu/connect_ideaa

Schema Evolution

What if the structure of the `people` table changes in `hoteldev`?

— In some cases Kafka will just handle it

— Select specific columns, not `select *`

https://z.umn.edu/connect_ideaa

Resources

— Kafka Connect Deep Dive – JDBC Source Connector

— Kafka Connect: Strategies To Handle Updates and Deletes

— Streaming data from Oracle into Kafka

— Kafka Connect 101: Introduction to Kafka Connect

— JDBC Source Connector: What could go wrong?

— Integrating Oracle and Kafka

— From Zero to Hero with Kafka Connect

https://z.umn.edu/connect_ideaa

Presentation Links

- These slides
  - https://z.umn.edu/connect_ideaa
- The demo video
  - https://z.umn.edu/connect_ideaa_demo
- Me
  - whit0694@umn.edu/Ian Whitney on OIT and Tech People Slacks
  - It is literally my job to talk about Kafka