

2023-04-25: foss-north 2023 (Gothenburg, Sweden)

# HOW TO TRANSFORM AVRO (IDL) DATA TO MULTIPLE PARQUET FILES

PANDÖÖRA

# AGENDA



- About Pandora and me (very shortly)
- Matching of expectations
- Background
- Proposed solution, FOSS and demo
- Summary
- Q&A

# ABOUT PANDORA AND ME (VERY SHORTLY)





# ABOUT PANDORA AND ME (VERY SHORTLY)

Pandora, world's biggest jewellery brand, 40-year anniversary last year summary:

- **1982 founded as small goldsmithing shop in CPH**
- 1989 crafting jewelry in Thailand
- **2000 signature charm bracelet concept is launched**
- 2005 fully-owned crafting facility opens in Bangkok
- 2014 strategic 10-year alliance with Disney (Star Wars, Marvel, ...)
- 2019 UNICEF partnership, donating USD 10 million, helping more than 1.2 million children
- **2020 crafting facilities running 100% renewable energy (CO2 neutral by 2025)**
- **2021 introduce jewelry with lab-created diamonds**
- 2022 employee number 32.000

- Our **values**:
  - WE DREAM
  - WE DARE
  - WE CARE
  - WE DELIVER



## ABOUT PANDORA AND **ME** (VERY SHORTLY)

- Lead Tech/Architect at Pandora's Digital Hub in CPH
  - Previous talks at foss-north:
    - 2020 - [DDD with Algebraic Data Types \(ADT\)](#)
    - 2019 - [Limiting side-effects of app at compile-time](#)
  - **Datology/datologist** are terms coined to **Peter Naur** (\*) and heavily inspired by (his friend) Edsger Dijkstra's quote: «**Computer Science is no more about computers than astronomy is about telescopes**», for when he founded the first Computer Science Department in Denmark (CPH University)
- (\*) - Only Dane with an ACM Turing Award

# MATCHING OF EXPECTATIONS



**CRAFT**  
THE *Incredible*

# MATCHING OF EXPECTATIONS

- Showcase a **tool** that **automates the transformation** of Apache Avro (**AVRO**), with **nested data**, into **multiple** Apache Parquet (**PARQUET**) **files, merging raw and structured/validated/deduplicated data** together
- Our **assumption** is that this allows for a **reduction** in the amount of **data storage and costs, by merging**, if we think in medallion lake-house architecture, the **bronze and silver layers**
- Furthermore, thanks to the **reduction in data storage**, it could also **help reduce CO2 emissions by roughly 33%**, using **fewer servers and less computing power**. The exact calculation and emissions reduction is **to be confirmed by the Pandora's Climate Team and Microsoft**, who are our cloud provider
- The **tool** is a free and open-source software (**FOSS**) library that is **available on NuGet**, and the source code is available at **Pandora's GitHub profile**

**Remark:** I would ♥ questions, but please save them to the end of the talk. Lots to say and time is mana, I mean limited

# BACKGROUND

# CRAFT

THE *Incredible*



# BACKGROUND

## KAFKA & DDD DATA MESH

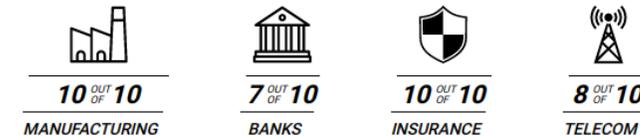
- We are **onboarding** Apache **Kafka** (KAFKA), a distributed event store and stream-processing platform that is an open-source system developed by the **Apache Software Foundation**. One of its most important capabilities is to provide scalable real-time integrations. It's **widely used** by the **biggest companies**
- Due to this and based on our **data mesh strategy**, our **domain-driven design (DDD) "ubiquitous"/common language** has become **AVRO IDL (subset of AVRO)**, as the format allows our: subject-matter experts (SME) & domain-experts; development & products teams; code and other stakeholders to have a **shared mental and common model** of the **representation of our data**
- Visit Confluent's blog post: [Saxo Bank's Best Practices for a Distributed Domain-Driven Architecture Founded on the Data Mesh](#) for info on DDD + Data Mesh

Source: <https://kafka.apache.org/>

### APACHE KAFKA

More than **80% of all Fortune 100 companies** trust, and use Kafka.

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.



[SEE FULL LIST](#)

10/10 Largest insurance companies  
10/10 Largest manufacturing companies  
10/10 Largest information technology and services companies  
8/10 Largest telecommunications companies  
8/10 Largest transportation companies  
7/10 Largest retail companies  
7/10 Largest banks and finance companies  
6/10 Largest energy and utilities organizations

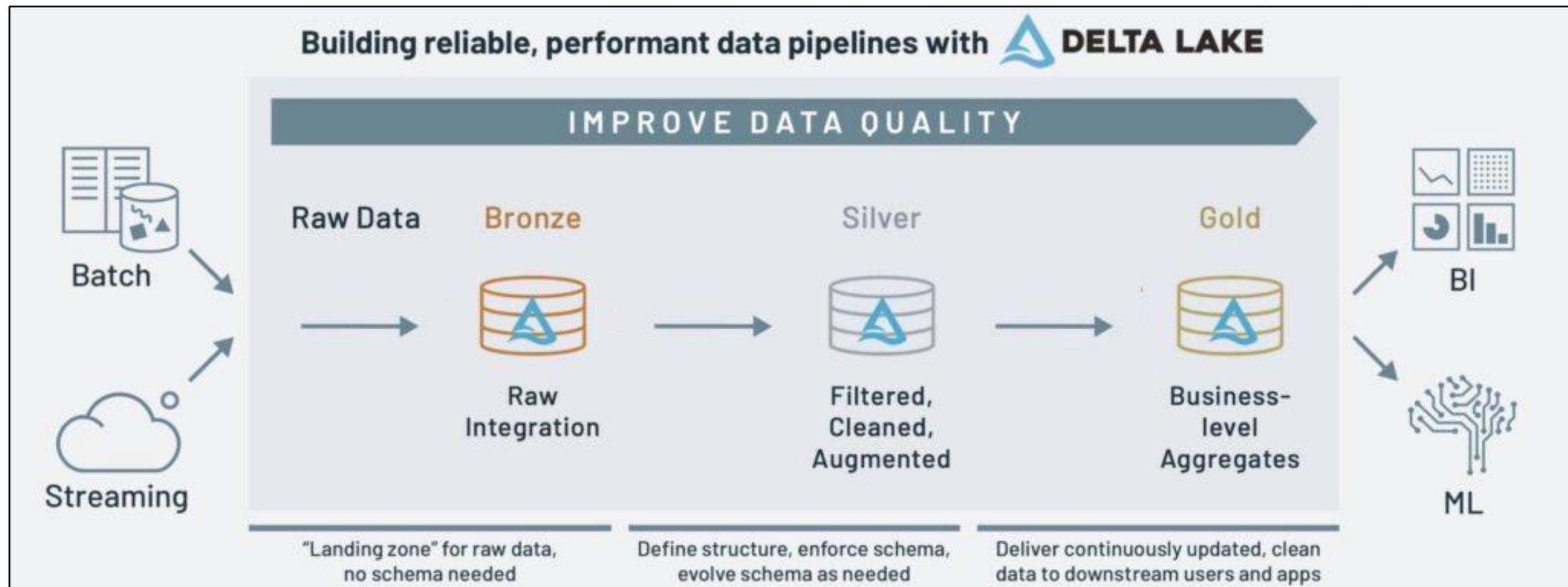
# BACKGROUND

## KAFKA (EDA) & DATA MESH DATA-LAKES (AVRO VS PARQUET)

- By using **KAFKA**, we will **rely on integrations** from **system-to-KAFKA** and **KAFKA-to-system** and hereby **reduce** to an **absolute minimum the system-to-system** integrations. This will ensure that **we do NOT expand our current systems** with **technical debt** just because we are going to **consume a new data stream**
- Since **KAFKA** is based on an **event-driven architecture (EDA)**, this would result in that **not ALL the data**, that is stored as an event in the EDA, **will be utilized by a given system**. And based on **the nature of EDA**, where the **important part** is the **communication strategy**, the **data will only be available** for a **short period of time**
- Therefore, it's **mandatory to introduce Event Sourcing (ES) as well**, where the **important part** is the **persistence strategy**, as that will allow us to **store ALL the data from the EDA** into our **data mesh data-lakes**
- To **make the data usable** in the **data mesh data-lakes**, the **AVRO data format might not be ideal** and therefore we would need to **transform it** to a format, that is optimized **to be consumed by the tools that normally are used by our data teams**
- The **PARQUET file format seems to be the de-facto standard** when it comes to persistence of data in data-lakes (Hadoop, Spark, Databricks, Synapse, ...). It is a column-oriented data storage format, which seems to differ from some of the other normally used formats (CSV, TSV, ...) and it **provides efficient data compression and encoding schemes** with **enhanced performance to handle complex data** in bulk

# BACKGROUND

## MEDALLION LAKE-HOUSE ARCHITECTURE (AVRO AS PARQUET)



Built-in Spark/Databricks (medallion lake-house architecture) will persist AVRO as single PARQUET file in bronze

Source: <https://www.databricks.com/glossary/medallion-architecture>

# BACKGROUND

## USING BUILT-IN TOOLING IS TEDIOUS AND COSTLY

- Spark/Databricks **built-in** AVRO to PARQUET (**persistence capabilities**):
  - <https://docs.databricks.com/external-data/avro.html>
  - **Main issues** by having AVRO nested data **stored in a single PARQUET file** is that it's **tedious and costly to work with** afterwards as you must **manually unwrap to reach specific data**
- Spark/Databricks **built-in** read and write to streamed AVRO (**real-time capabilities**):
  - <https://docs.databricks.com/structured-streaming/avro-dataframe.html>
  - **Main issues** with this approach is that a **cluster must be running 24/7** and the **same constraints (working with nested data)** as mentioned in previous bullet-point, apply as well
    - **Azure Databricks cluster: 400 USD/month** (Standard light) - **925 USD/month** (Premium)
      - Source: <https://cprosenjit.medium.com/azure-databricks-cost-optimizations-5e1e17b39125>
    - **Azure K8S container: 15 EUR/month** and **Azure Container App: 17 EUR/month**
      - Source: <https://jussiroyne.com/2021/12/running-a-single-docker-container-in-azure-cost-effectively>

# PROPOSED SOLUTION, FOSS AND DEMO



**CRAFT**  
THE *Incredible*

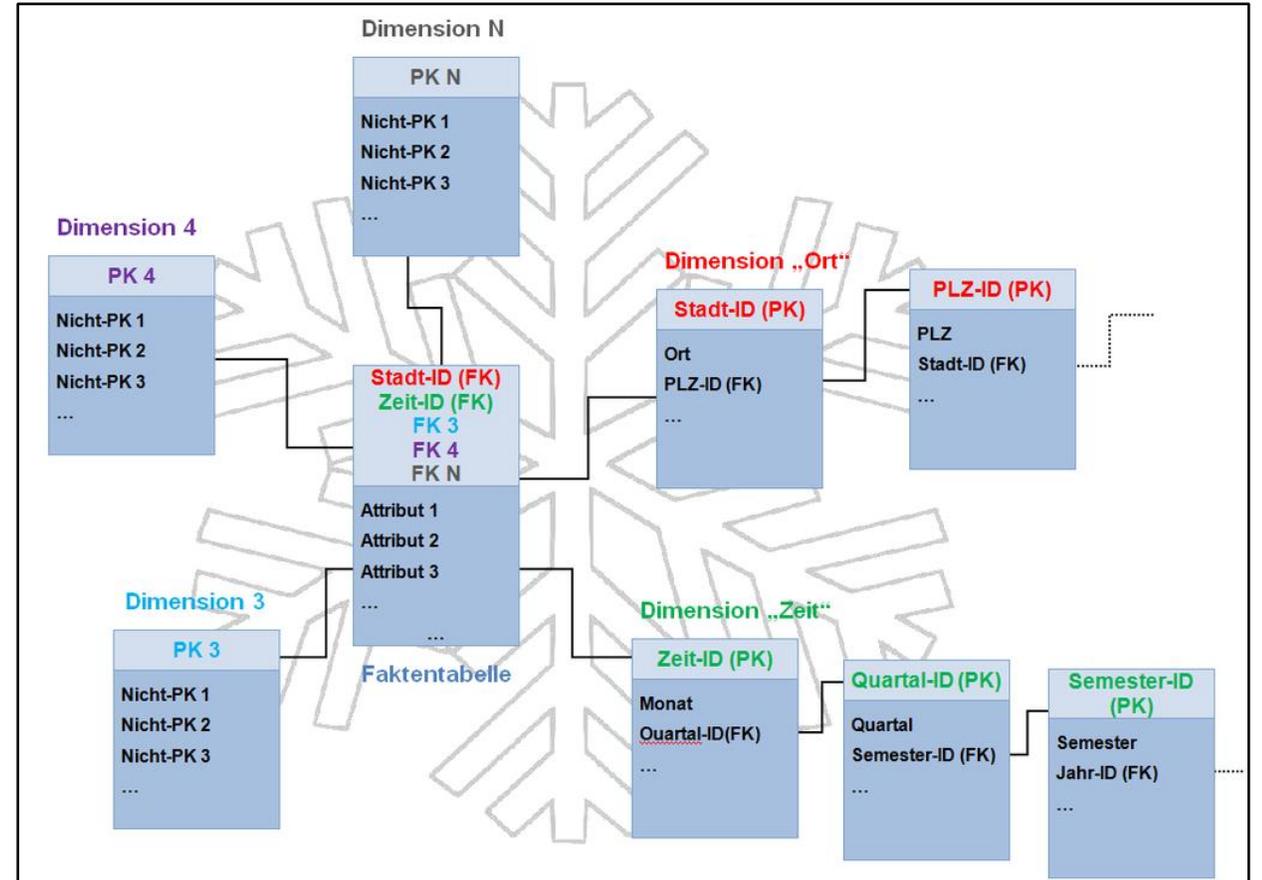
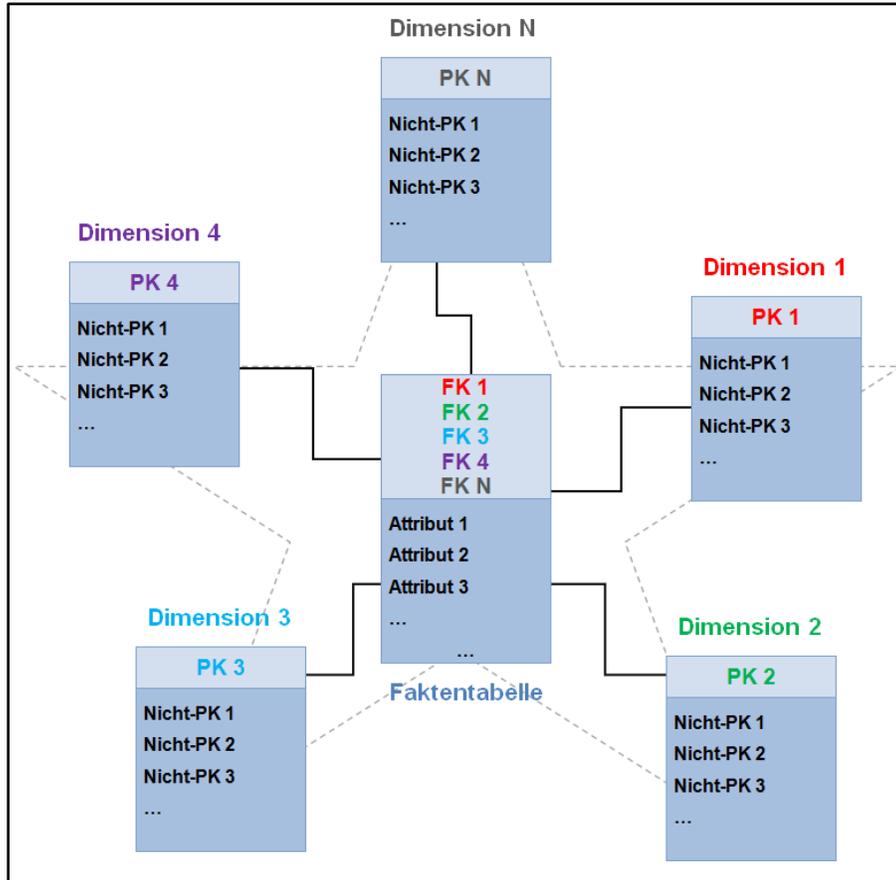
# PROPOSED SOLUTION, FOSS AND DEMO

## PROPOSED SOLUTION > TOOL

- The tool presents a **novel and state-of-the-art solution** and **transforms** our Apache Avro® data (**AVRO IDL**) into multiple Apache Parquet® (**PARQUET**) files in a fully automated way
- **AVRO** and **PARQUET** are formats to **handle data in a very-optimized and cost-effective manner** and are **widely accepted and used**
- This means **nested data elements** are **shown as an extension table** (a separate file) and **allows us to merge raw and validated/deduplicated** data together, **making it easier** for data engineers/scientists and business analysts **to utilize data**

# PROPOSED SOLUTION, FOSS AND DEMO

## PROPOSED SOLUTION > TOOL > EXTENSION TABLES



Fact and dimension tables (star schema) and recursively (snowflake schema)

Source: [https://en.wikipedia.org/wiki/Star\\_schema](https://en.wikipedia.org/wiki/Star_schema) and [https://en.wikipedia.org/wiki/Snowflake\\_schema](https://en.wikipedia.org/wiki/Snowflake_schema)

# PROPOSED SOLUTION, FOSS AND DEMO

## PROPOSED SOLUTION > TOOL > STAR/SNOWFLAKE SCHEMAS

- **Benefits** of Star/Snowflake schemas:
  - **Simpler queries**
  - **Simplified** business reporting logic
  - **Query performance gains**, when read-only (\*)
  - **Fast aggregations**
  - **Efficient** and **compact storage** of normalised data
  - Sample [Querying One Trillion Rows of Data with PowerBI and Azure Databricks](#):
    - Tables: 1 fact, 2 dimension and 1 aggregate
    - **Queries** only take a **few seconds**
    - **Delta Lake**

(\*) - Suits us well as **we are only atomically appending** the **immutable data stream of events** for **historical purposes**

# PROPOSED SOLUTION, FOSS AND DEMO

## PROPOSED SOLUTION > TOOL > TRANSFORM NESTED TYPES & SHA256

- Before we can automatically generate our Star/Snowflake schemas, we will **need to translate the nested** (and complex) **AVRO IDL types** to a **single and uniform type** that can handle them all. We can achieve this by doing the following transformations:
  - `array of 't` => `record { item : 't }`
  - `map of 't` => `record { key : string, value : 't }`
  - `union of 't seq` => `record { type0 : 't_0, ..., typeN : 't_n }`
- By doing this, we can now utilize the [Avro.Generic.GenericRecord](#) class to **transform any given AVRO IDL record type to our defined PARQUET schema AST**
- Furthermore, due to the **nature of AVRO IDL** and Application Lifecycle Management (**ALM**) where it is **supported to do changes to the schemas**, it's **important to ensure** that we **do NOT get invalid data in the data-lakes**. Therefore, we are performing a **SHA-256 hash** on the [Avro.Generic.GenericRecord](#) **Schema (string) property**, to ensure that **ALL the related data to that given schema version**, will go to the **right data-lake folder**. The generation of the SHA-256 hash is **always deterministic** in the sense if **v.1.0 and v.42.0 are equal**, then data will **share the same folder** ([surjective mapping](#))

# PROPOSED SOLUTION, FOSS AND DEMO

## PROPOSED SOLUTION > TOOL > DATA-LAKE VS DELTA LAKE (JSONL)

The image displays two screenshots of the Azure Storage Explorer interface, illustrating the structure of Delta Lake JSONL control files.

**Left Screenshot:** Shows a directory view with 7 items selected (7 selected). The items are:

- [-.]
- Foo
- Interop
- InteropMapField
- InteropUnionField
- InteropUnionFieldType2
- Node
- test

**Right Screenshot:** Shows a detailed view of 9 items, all JSONL control files. The items are:

Name	Last modified
[.]	
00000000000000000000000000000000.json	4/17/2023, 7:23:50 AM
00000000000000000000000000000001.json	4/17/2023, 7:24:02 AM
00000000000000000000000000000002.json	4/17/2023, 7:24:13 AM
00000000000000000000000000000003.json	4/17/2023, 7:24:25 AM
00000000000000000000000000000004.json	4/17/2023, 7:24:32 AM
00000000000000000000000000000005.json	4/17/2023, 7:24:44 AM
00000000000000000000000000000006.json	4/17/2023, 7:24:54 AM
00000000000000000000000000000007.json	4/17/2023, 7:30:19 AM
00000000000000000000000000000008.json	4/17/2023, 7:30:28 AM

Delta Lake JSONL control files provides fast & efficient queries and ignore/prune of irrelevant partitions capabilities

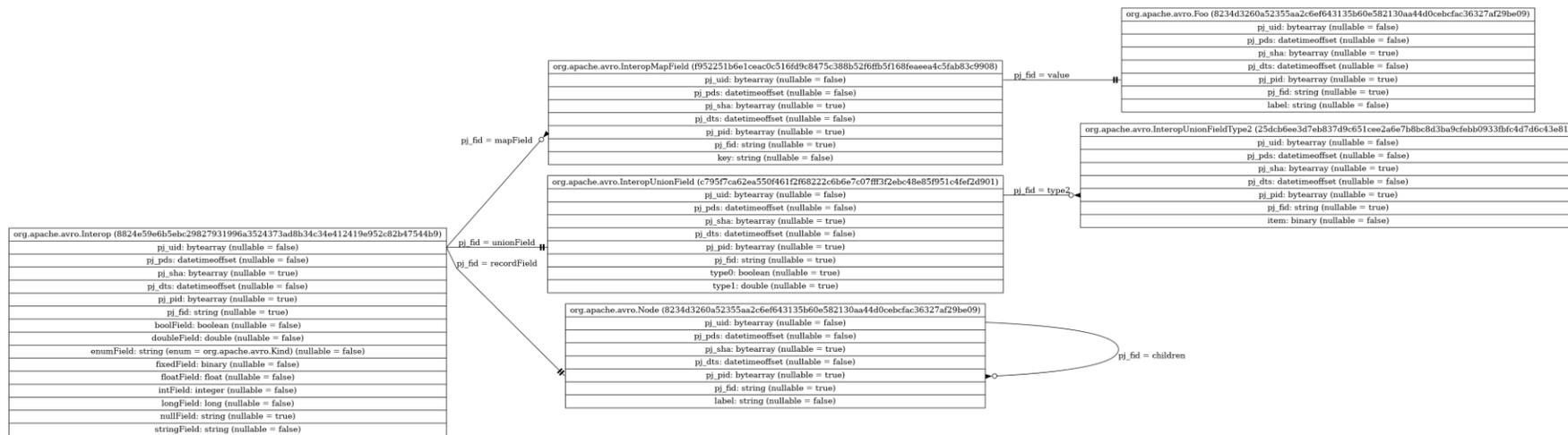
# PROPOSED SOLUTION, FOSS AND DEMO

## PROPOSED SOLUTION > TOOL > GOALS

- **Ripple effect:** Fully automated propagation of schemas to data storages, based on well-defined data-contracts
- **Onboarding:** Ease onboarding to KAFKA by providing 2-for-1 development (create producer, get consumer for free)
- **Delta Lake support:** This means we can search in massive datasets, very quickly. By delivering to a lake-house architecture that has support for data partitions, it will allow for fast and efficient queries of large datasets by using parallelization as well as quickly ignore/prune of irrelevant partitions. Furthermore, the Delta Lake is cloud agnostic
- **User friendliness (UI/UX):** Data engineers/scientists and business analysts will be able to keep using the tools they used too. It will be a fully transparent experience for them
- **Automation and cost & time reduction:** Adding our solution to a data pipeline will only require a few lines of configuration (no code needed). From that point, the consumed data, will be delivered and persisted to the chosen data storage reducing engineering tasks as well as cloud storage/processing
- **Our sustainability journey:** We believe high-quality (jewellery), strong business performance and high ethical standards go hand in hand, and we craft (our jewellery) with respect for resources, environment and people.

# PROPOSED SOLUTION, FOSS AND DEMO

## PROPOSED SOLUTION > TOOL > GOALS > RIPPLE-EFFECT



```

/**
 * Currently genavro only does Protocols.
 */
@namespace("org.apache.avro")
protocol InteropProtocol {

  record Foo {

    string label;

  }

  enum Kind {

    A,
    B,
    C

  }

  fixed MD5 (16);

  record Node {

    string label;

    array<org.apache.avro.Node> children = [];

  }

  record Interop {

    int intField = 1;

    long longField = -1;

    string stringField;

    boolean boolField = false;

    float floatField = 0.000000;

    double doubleField = -100000000000.000000;

    null nullField;

    map<org.apache.avro.Foo> mapField;

    union{boolean,double,array<bytes>} unionField;

    org.apache.avro.Kind enumField;

    org.apache.avro.MD5 fixedField;

    org.apache.avro.Node recordField;

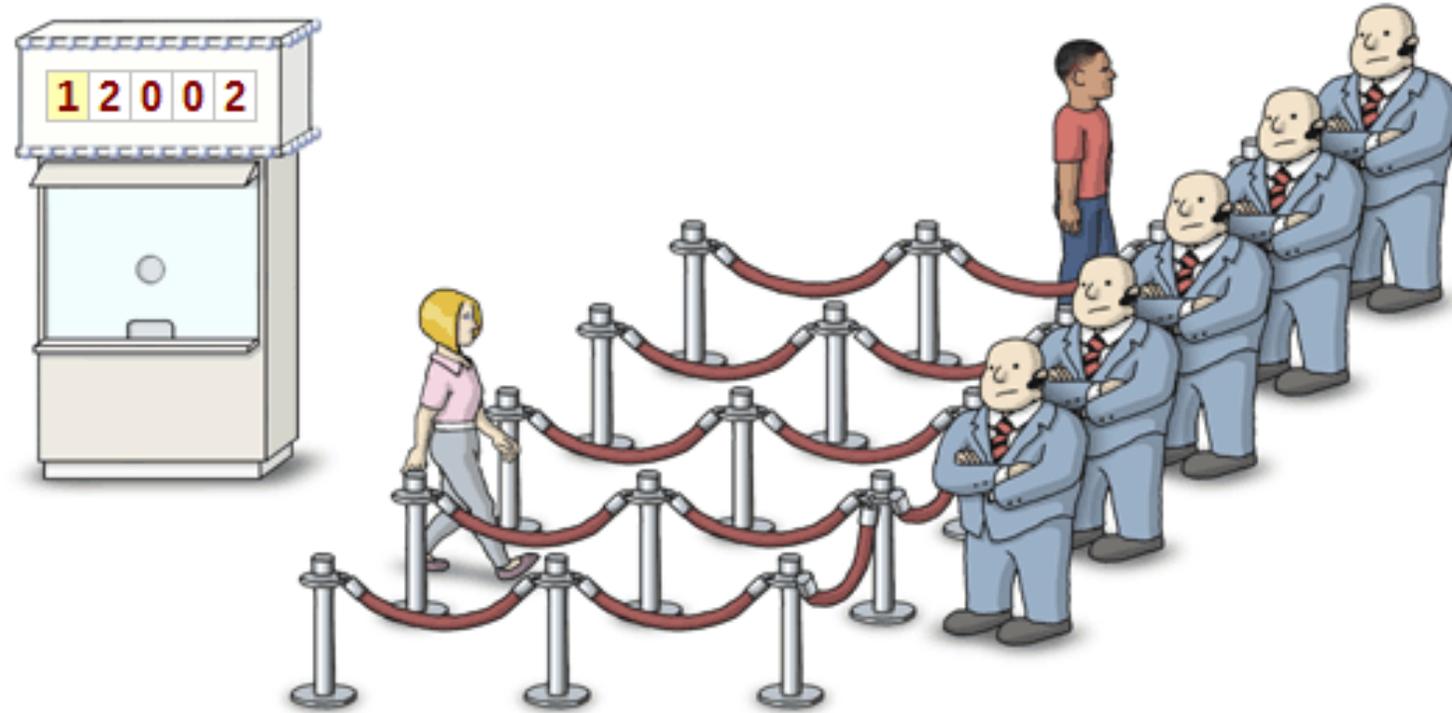
  }

}
  
```

Source: <https://github.com/apache/avro/blob/master/lang/java/compiler/src/test/idl/input/interop.avdl>

# PROPOSED SOLUTION, FOSS AND DEMO

PROPOSED SOLUTION > TOOL > GOALS > ONBOARDING

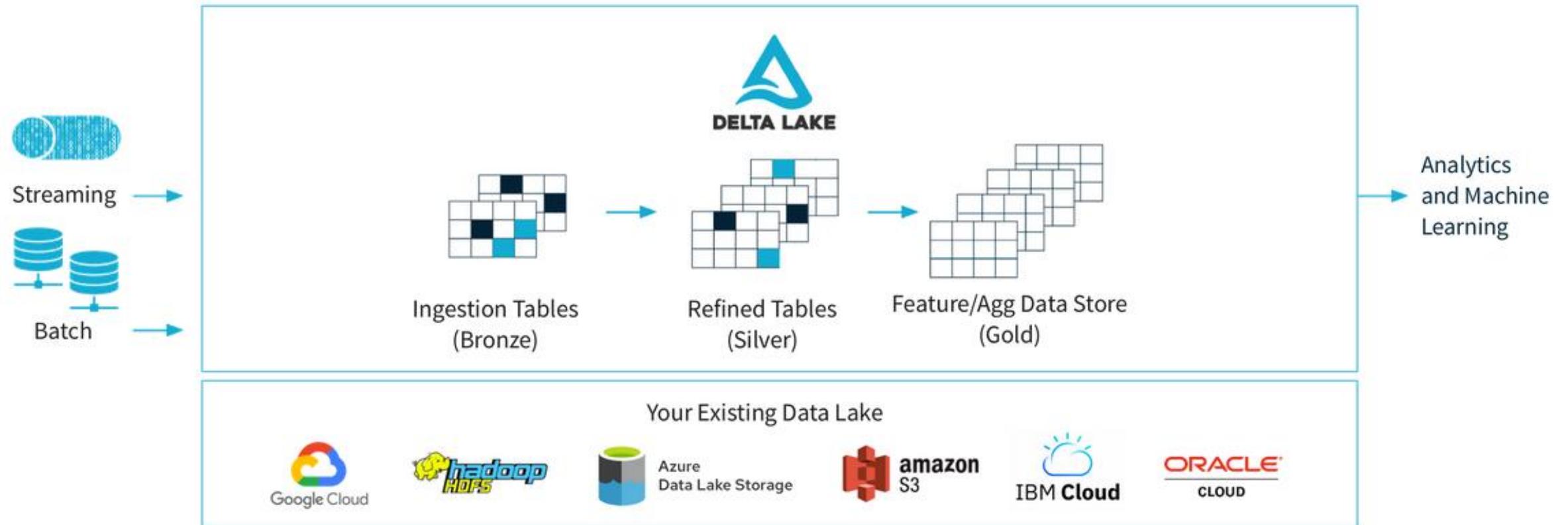


- **Invest** in creating a **data producer** that **ingests data to KAFKA**
- **Use generic sink/archiver consumer**, built with the **FOSS library**, to **deliver data to a specific data mesh data-lake in a buffered manner** (combination of streaming/batch) relying on both **time interval** and **buffer upper-bound semaphores**

Source: <https://preshing.com/20150316/semaphores-a-re-surprisingly-versatile/>

# PROPOSED SOLUTION, FOSS AND DEMO

PROPOSED SOLUTION > TOOL > GOALS > DELTA LAKE SUPPORT



Delta Lake is OSS (The Linux Foundation) and is cloud agnostic

Source: <https://delta.io/>

# PROPOSED SOLUTION, FOSS AND DEMO

## PROPOSED SOLUTION > TOOL > GOALS > WELL-KNOWN UI/UX

```
1 %sql
2
3 -- Azure Databricks Delta Lake:
4 --
5 -- https://[REDACTED].blob.core.windows.net/[REDACTED]/demo/delta/org/apache/avro/Interop/8824e59e6b5ebc29827931996a3524373ad8b34c34e412419e952c82b47544b9/
6
7 -- AVRO IDL Schema v.1.0
8 ( select distinct
9   * except
10  ( pj_uid
11    , pj_pds
12    -- , pj_dts
13    , pj_pid
14    , pj_fid
15  )
16  from
17  delta.`/mnt/adl2/[REDACTED]/demo/delta/org/apache/avro/Interop/8824e59e6b5ebc29827931996a3524373ad8b34c34e412419e952c82b47544b9/`
18  where
19  pj_pds >= pj_timestamp() -- '2023-04-17'
20  order by pj_dts desc
21  limit 10
22 )
23 -- AVRO IDL Schema v.x.0
24 union
25 ( select distinct
26   * except
27  ( pj_uid
28    , pj_pds
29    -- , pj_dts
30    , pj_pid
31    , pj_fid
32  )
33  from
34  delta.`/mnt/adl2/[REDACTED]/demo/delta/org/apache/avro/Interop/8824e59e6b5ebc29827931996a3524373ad8b34c34e412419e952c82b47544b9/`
35  where
36  pj_pds >= pj_timestamp() -- '2023-04-17'
37  order by pj_dts desc
38  limit 10
39 )
```

▶ (3) Spark Jobs

▶ \_sqlidf: pyspark.sql.dataframe.DataFrame = [pj\_sha: binary, pj\_dts: timestamp ... 9 more fields]

Data scientists and business analysts are used to Databricks python notebook & SQL cells as well as Azure Synapse SQL

# PROPOSED SOLUTION, FOSS AND DEMO

## PROPOSED SOLUTION > TOOL > GOALS > AUTOMATION + COST & TIME

- **Less storage and CPU-cycles** at cloud providers by **merging bronze and silver** layers
  - **Bronze + silver**: full, raw and history of each dataset combined with defined structure, enforced schemas as well validated and deduplicated data
  - **Gold**: data as knowledge
- **Fewer engineering resources** that otherwise **would manually (\*) define pipelines to transform data** between layers

(\*) - Sample of data-flow that most of us will recognize right?

1. Move data from source (landing) to raw
2. Move from raw to enriched
3. Move from enriched to curated
4. Move from curated to development & archive
5. Create views
6. Visualize created views with reporting tool

# PROPOSED SOLUTION, FOSS AND DEMO

## PROPOSED SOLUTION > TOOL > GOALS > SUSTAINABILITY

- Pandora wants to be a **sustainability leader**:
  - **100% renewable energy** (CO2 neutral by 2025)
  - **Recycled gold and silver**
  - **Lab-grown diamonds** (and ditching mined diamonds):
    - CNN Business: [The world's largest jewelry brand is ditching mined diamonds](#)
  - More **info** at **yearly sustainability reports**:
    - <https://pandoragroup.com/sustainability/resources/sustainability-reports>
- By **merging two medallion layers** into a single, **could lead** to the **possible savings of a 1/3 in disk usage**
- We are **NOT relying on a naive approach**, when flattening and storing data, therefore, it **could lead to greater savings**
- **Providing the tool for others** to follow in our steps, **could have an impact on a global scale**

**Remark:** The exact **calculation and emissions reduction** is **to be confirmed by the Pandora's Climate Team and Microsoft**, who are our cloud provider

# PROPOSED SOLUTION, FOSS AND DEMO

## FOSS > LGPL

### GNU LGPLv3

Permissions of this copyleft license are conditioned on making available complete source code of licensed works and modifications under the same license or the GNU GPLv3. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights. However, a larger work using the licensed work through interfaces provided by the licensed work may be distributed under different terms and without source code for the larger work.

Permissions	Conditions	Limitations
<ul style="list-style-type: none"><li>Commercial use</li><li>Distribution</li><li>Modification</li><li>Patent use</li><li>Private use</li></ul>	<ul style="list-style-type: none"><li>Disclose source</li><li>License and copyright notice</li><li>Same license (library)</li><li>State changes</li></ul>	<ul style="list-style-type: none"><li>Liability</li><li>Warranty</li></ul>

---

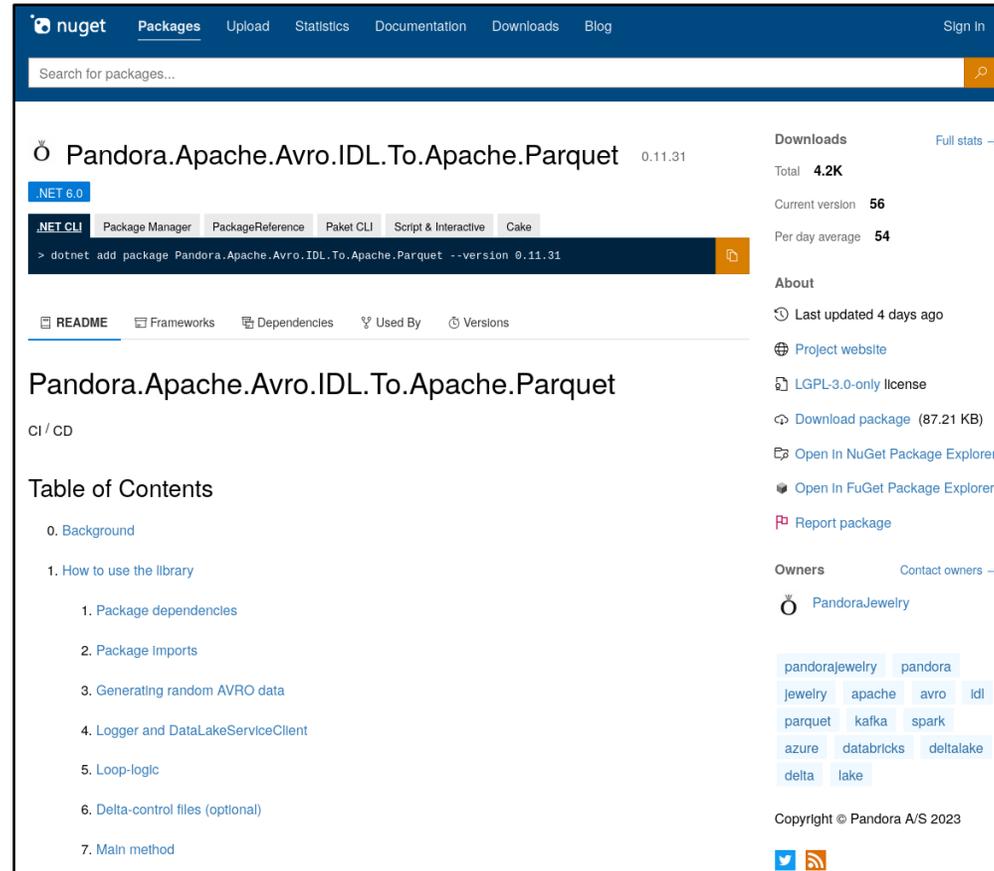
[View full GNU Lesser General Public License v3.0 »](#)

**tl;dr:** Permissive FOSS that allows you to use the library (at your own risk). You aren't forced to open-source your related work. However, if you make any changes to the library itself, please share it with the rest of the community.

Source: <https://choosealicense.com/licenses/>

# PROPOSED SOLUTION, FOSS AND DEMO

## FOSS > NUGET



The screenshot shows the NuGet package page for `Pandora.Apache.Avro.IDL.To.Apache.Parquet` version 0.11.31. The page includes a search bar, navigation links (Packages, Upload, Statistics, Documentation, Downloads, Blog), and a sign-in option. The package details section shows the current version (0.11.31), total downloads (4.2K), and current version count (56). The package is available for .NET 6.0 and .NET CLI. The package manager is Package Manager, and the package reference is `Paket CLI`. The package is used by `Script & Interactive` and `Cake`. The package is licensed under LGPL-3.0-only. The package is available for download (87.21 KB). The package is available in NuGet Package Explorer and FuGet Package Explorer. The package is reported by PandoraJewelry. The package is owned by PandoraJewelry. The package is available for download (87.21 KB). The package is available in NuGet Package Explorer and FuGet Package Explorer. The package is reported by PandoraJewelry. The package is owned by PandoraJewelry. The package is available for download (87.21 KB). The package is available in NuGet Package Explorer and FuGet Package Explorer. The package is reported by PandoraJewelry. The package is owned by PandoraJewelry.

**Pandora.Apache.Avro.IDL.To.Apache.Parquet** 0.11.31

.NET 6.0

.NET CLI Package Manager PackageReference Paket CLI Script & Interactive Cake

```
> dotnet add package Pandora.Apache.Avro.IDL.To.Apache.Parquet --version 0.11.31
```

README Frameworks Dependencies Used By Versions

**Pandora.Apache.Avro.IDL.To.Apache.Parquet**

CI / CD

Table of Contents

- 0. Background
- 1. How to use the library
  - 1. Package dependencies
  - 2. Package Imports
  - 3. Generating random AVRO data
  - 4. Logger and DataLakeServiceClient
  - 5. Loop-logic
  - 6. Delta-control files (optional)
  - 7. Main method

Downloads [Full stats →](#)

Total **4.2K**

Current version **56**

Per day average **54**

About

Last updated 4 days ago

[Project website](#)

[LGPL-3.0-only license](#)

[Download package \(87.21 KB\)](#)

[Open in NuGet Package Explorer](#)

[Open in FuGet Package Explorer](#)

[Report package](#)

Owners [Contact owners →](#)

[PandoraJewelry](#)

[pandorajewelry](#) [pandora](#) [jewelry](#) [apache](#) [avro](#) [idl](#) [parquet](#) [kafka](#) [spark](#) [azure](#) [databricks](#) [deltalake](#) [delta](#) [lake](#)

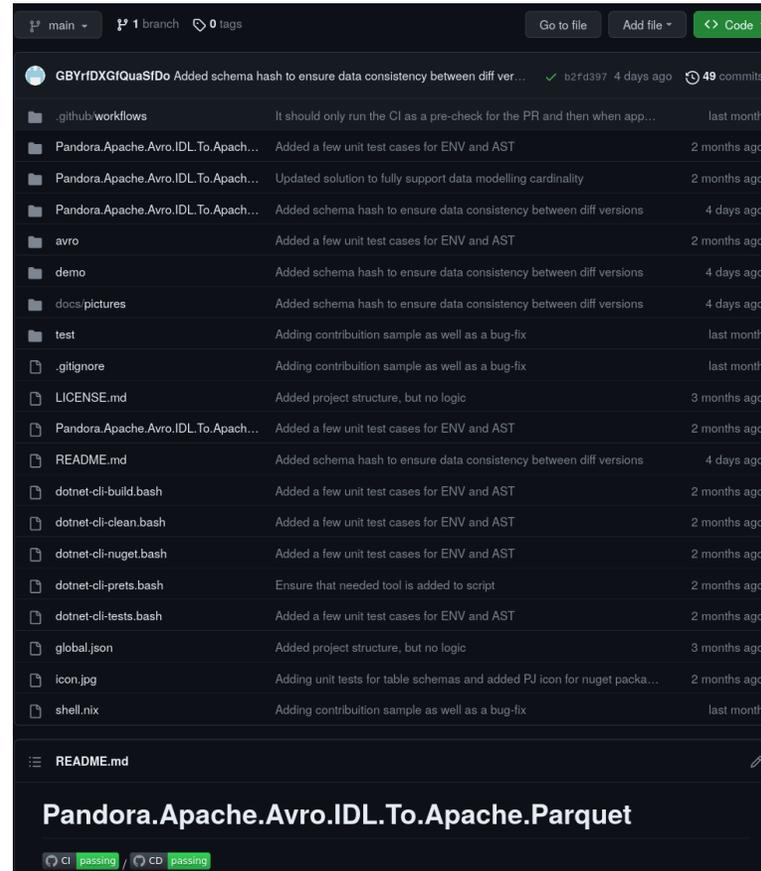
Copyright © Pandora A/S 2023

[Twitter](#) [RSS](#)

Library available at NuGet: <https://www.nuget.org/packages/Pandora.Apache.Avro.IDL.To.Apache.Parquet>

# PROPOSED SOLUTION, FOSS AND DEMO

## FOSS > GITHUB



Source code available on GitHub: <https://github.com/PandoraJewelry/Pandora.Apache.Avro.IDL.To.Apache.Parquet>

# PROPOSED SOLUTION, FOSS AND DEMO

## DEMO > ER-DIAGRAMS, WITH THE CARDINALITY, BETWEEN TABLES

- Demo steps (based on **avroidl2dot** script):
  1. Emphasis on consuming the public NuGet package: **Pandora.Apache.Avro.IDL.To.Apache.Parquet**
  2. Run **avroidl2dot**, show console log and content of the **./dots/** folder
  3. Show both the generated **PNG** and **SVG** files for the Interop **AVRO IDL** file:

Source: <https://github.com/apache/avro/blob/master/lang/java/compiler/src/test/idl/input/interop.avdl>

# SUMMARY



# SUMMARY



- Matching of expectations
- Background
- Proposed solution, FOSS and demo
- Q&A

Our values: We dream, dare, care and deliver

Q&A

# CRAFT

THE *Incredible*

