

Airport for DuckDB:

Letting DuckDB take Apache Arrow Flights

Rusty Conover - rusty@conover.me - DuckCon #6, Amsterdam, 2025-01-31

DuckDB Extensions I've created

- `Crypto` - Cryptographic hash functions
- `Datasketches` - Probabilistic data structures
- `Evalexpr_rhai` - Embedded scripting language
- `Fuzzycomplete` - Alternative autocompletion in CLI
- `Lindel` - Linearization (Morton/Hilbert curves)
- `Shellfs` - Subprocess I/O

Imagine a world where...

DuckDB makes tabular data access
effortless and **universal**.

Any data, anywhere, on all systems.

All via SQL.

What is the Airport Extension?

An explanation by analogy.

Airports let you go to far away places

Flying is quick and efficient

You can bring things back

You can leave things there



What is the Airport Extension?

An explanation by analogy.

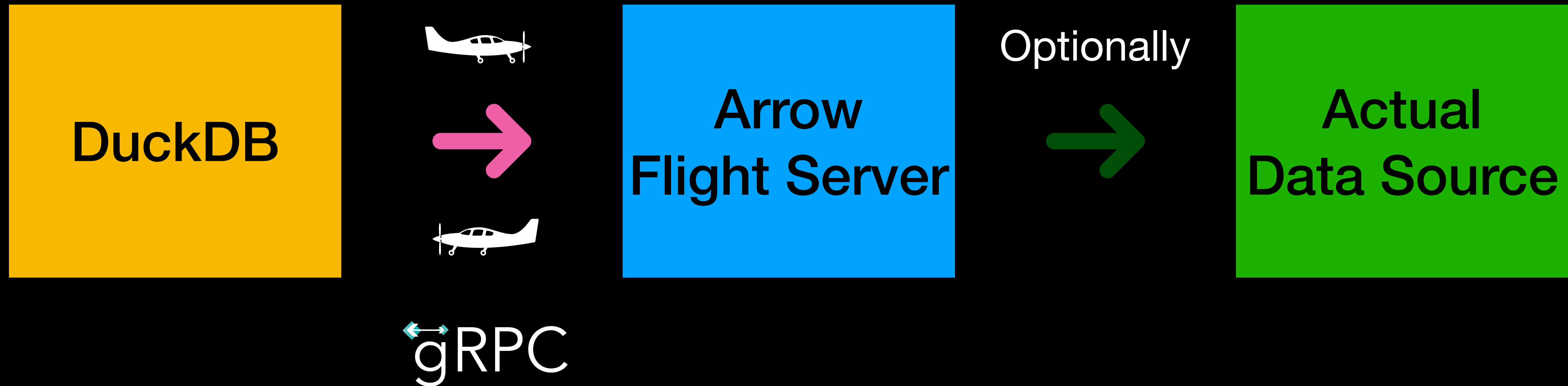
Airports let you go to far away places	Remote Servers and Systems
Flying is quick and efficient	Uses Apache Arrow Flight
You can bring things back	SELECT statements
You can leave things there	INSERT, UPDATE, DELETE...

Where can I fly to?

Anywhere sunny and somewhat tabular.

- “Lakehouse” Formats - Iceberg, Delta Lake, Vortex, Hudi, LanceDB, Nimble
- NoSQL - Redis, DynamoDB, MongoDB, Cassandra, AirTable
- Graph Databases - Kuzu, Neptune, Neo4j
- ★DuckDB instances★ on other machines
- Spark - Spark Connect
- DataBricks - Delta Sharing
- REST APIs: Stripe, Shopify, Github
- Event Busses and Queues: Kafka, WarpStream, SQS, RabbitMQ
- Old School Databases: LDAP, DNS, LMDB
- Other SQL Servers: SQL Server, Oracle, SQLite, Limbo
- Cloud Management: AWS, Google Cloud
- Management Services: Kubernetes, SNMP, Routing Tables
- Legacy Systems: Of course

How does Arrow Flight Work



Why did I create the Airport extension

Benefits

- Less code for new “extensions”.
- Leverage programming language ecosystems (PyPi, Crates)
- Runs outside of DuckDB
 - Complexity contained, Responsibility is clear, No crashes
 - Access distributed hardware and resources
- Simplifies the build, test, distribute loop
- Can offer data-as-a-service and function-as-a-service

Ready for some demos?

Delta Lake with Write Support

AutoGluon: Machine Learning with only SQL

Maybe: Geocoding and Weather



Delta Lake With Write Support



```
ATTACH 'deltalake' (  
  TYPE AIRPORT,  
  location 'grpc://localhost:50312/'  
);  
CREATE SCHEMA deltalake.test1;  
CREATE TABLE deltalake.test1.people (  
  name VARCHAR,  
  love_of_duckdb INT,  
  tags VARCHAR[]  
);
```

Delta Lake With Write Support



```
INSERT INTO deltalake.test1.people values  
(  
  'rusty', 5, ['airport', 'datasketches']),  
(  
  'sam', 10, ['deltalake', 'iceberg']);
```

```
SELECT * FROM deltalake.test1.people;  
      name = rusty  
love_of_duckdb = 5  
      tags = [airport, datasketches]  
  
      name = sam  
love_of_duckdb = 10  
      tags = [deltalake, iceberg]
```


Delta Lake With Write Support

Yes this could be S3, Azure, GCS, R2...

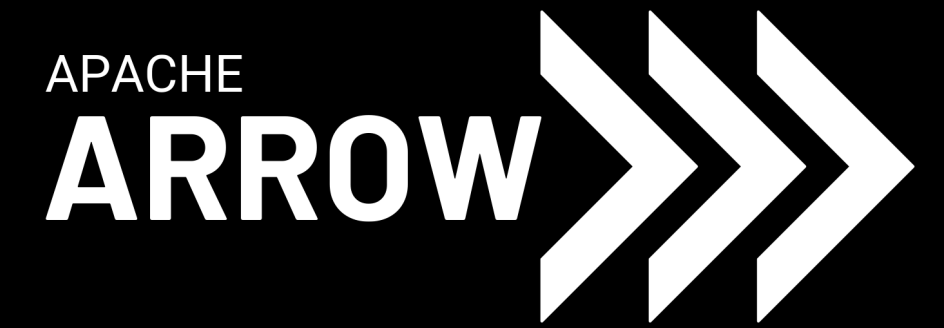


```
$ ls -lR
total 8
drwxr-xr-x  4 rusty  staff   128 Jan 30 23:13 _delta_log
-rw-r--r--  1 rusty  staff  1225 Jan 30 23:13 part-00001-82e3eb4e-
b4e1-4344-ad1d-4c9183c918e8-c000.snappy.parquet

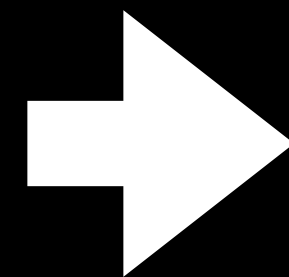
./_delta_log:
total 16
-rw-r--r--  1 rusty  staff  1586 Jan 30 23:12 00000000000000000000000000000000.json
-rw-r--r--  1 rusty  staff   737 Jan 30 23:13 00000000000000000000000000000001.json
```


Delta Lake With Write Support

How it works



DuckDB



Arrow
Flight Server
Python delta-rs

Predicate
Pushdown

DuckDB
Catalog
Integration



In the interest of time, I'm skipping over:

Time Travel

Row Change
Tracking

Run jobs
outside of
DataBricks

Compaction /
Partitions

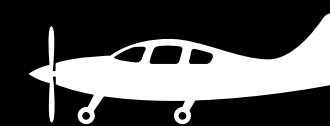
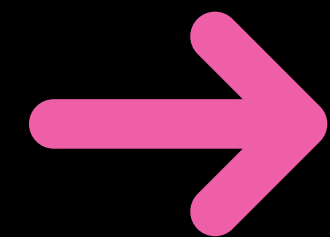
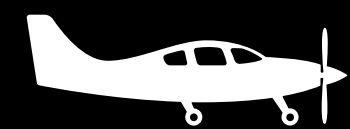
Unity
Catalog
Integration

dbt

How does Arrow Flight Work

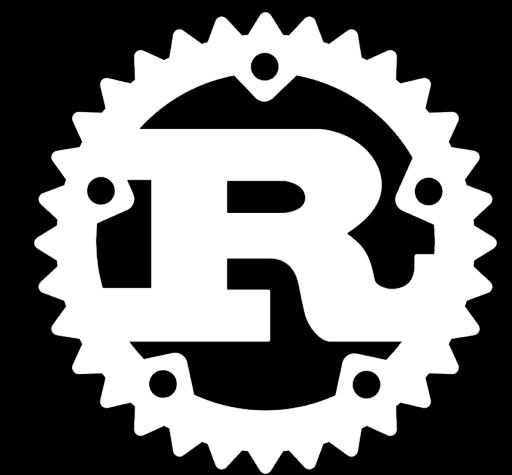
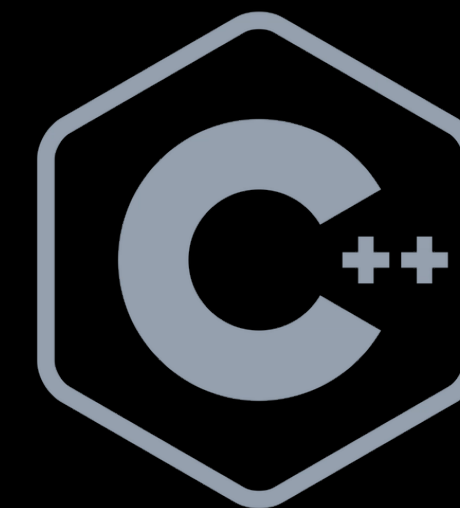


DuckDB

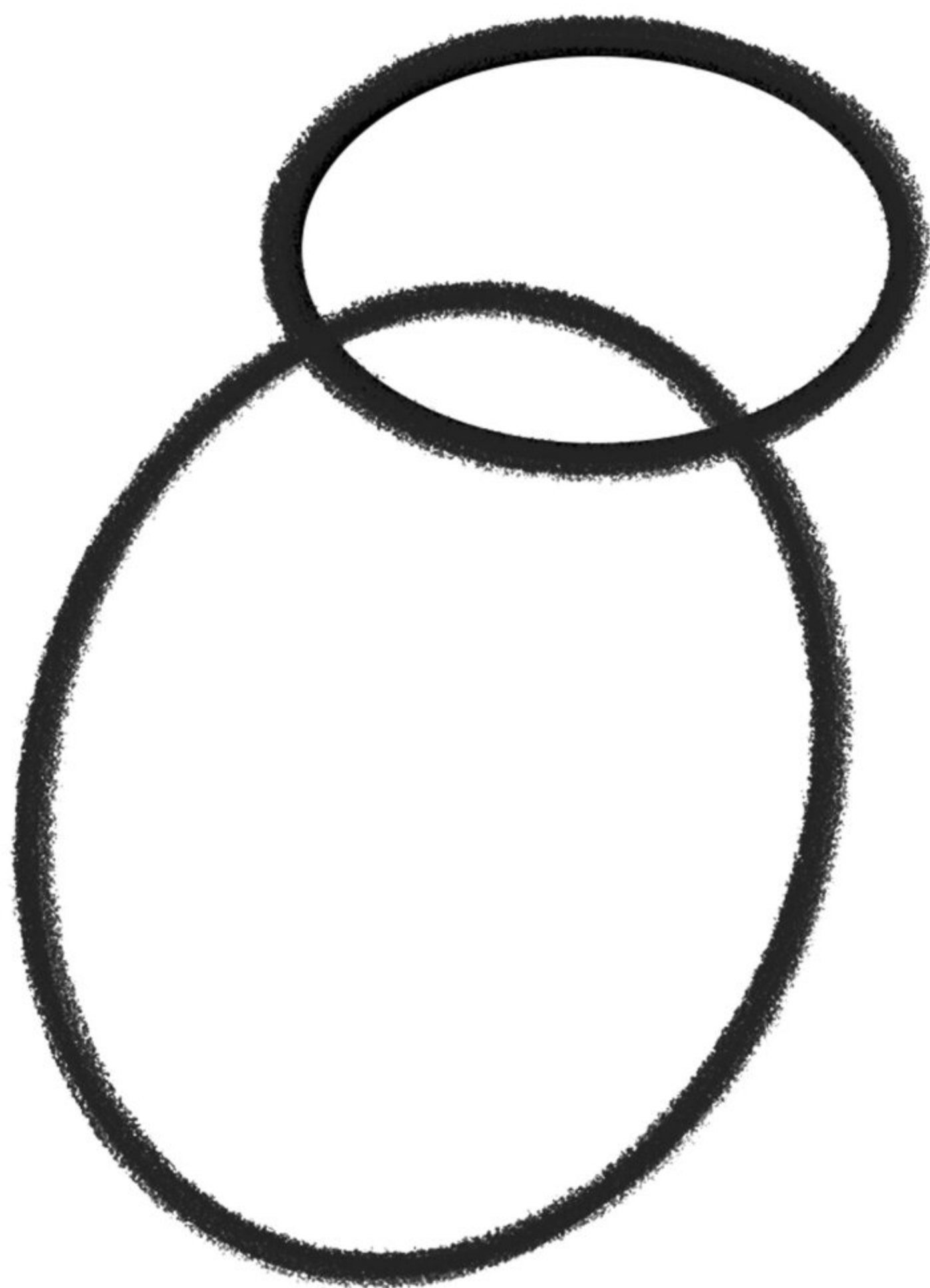


Process Boundary

Arrow
Flight Server



- Can be written in Java, Python, Rust, C#, C++
- Runs out of process with DuckDB, no need to relink/rebuild/distribute dependencies.
- 🕒 Skipping details due to time.



Step 1: Draw some circles



Step 2: Draw the rest of the owl!

AutoGluon

“Fast and Accurate ML in 3 Lines of Code”

- Build a ML Model with A Few **SELECT** Statements.
- Tabular Prediction
 - Binary/Multiclass Classification, Regression, Quantile Prediction
- Time Series Prediction
- No need to get into a Python notebook



Hacker News Votes Prediction



```
ATTACH 'autogluon' (TYPE AIRPORT, location  
'grpc://localhost:50312/');
```

```
CREATE SCHEMA autogluon.p1;
```

```
CREATE TABLE  
  autogluon.p1.hn_stories as  
  SELECT  
    title,  
    to_timestamp(time) as post_time,  
    score::float as score  
  FROM 'hacker-news-stories.parquet';
```


Hacker News Votes Prediction



Create the model, “fit the predictor”...

```
SELECT * FROM
autogluon.p1.predictor_fit(
    'hn_votes',      - model name
    'hn_stories',    - training data
    'score',         - target column
    problem_type='regression',
    time_limit=200,
    presets='high_quality')
```

Hacker News Votes Prediction



Create some example data for the model to use.

```
CREATE TABLE example_headlines (title text);

INSERT INTO example_headlines values
  ('DuckDB 1.2.0'),
  ('Iceberg versus Delta Lake, tales from the
format war'),
  ('SQL tips and tricks'),
  ('AI will replace all CS graduates'),
  ('Spaces are better than tabs, prove me wrong');
```

Hacker News Votes Prediction



Calling the model to make predictions

```
SELECT title, prediction
FROM
autogluon.p1.predictor_predict_rows(
    'hn_votes',          -- model name
    (SELECT title,       -- model input
     now() as post_time
    FROM example_headlines));
```


Hacker News Votes Prediction



Prediction Results

```
title = DuckDB 1.2.0  
prediction = 34.16758
```

```
title = Iceberg versus Delta Lake, tales from the format war  
prediction = 25.366014
```

```
title = SQL tips and tricks  
prediction = 16.140633
```

```
title = AI will replace all CS graduates  
prediction = 13.036682
```

```
title = Spaces are better than tabs, prove me wrong  
prediction = 38.696304
```


Hacker News Votes Prediction



Testing HN post titles

```
SELECT * FROM
  autogluon.p1.predictor_predict_rows('hn_votes',
    (SELECT
      'Airport Extension for DuckDB using Arrow Flight' as
    title,
      now() as post_time
    ));

  title = Airport Extension for DuckDB using Arrow Flight
post_time = 2025-01-31 08:06:40.764+00
prediction = 12.30423
```

Geocoding and Weather (Bonus Content)

```
ATTACH 'geocoder' (  
    TYPE AIRPORT, location 'grpc://localhost:50212/'  
);
```

```
SELECT geocoder.usa.geocode_address(  
    '1600 Pennsylvania Ave, Washington, DC'  
) as result;
```

```
result = {  
    'latitude': 38.879389288728,  
    'longitude': -76.982767739978  
}
```

Geocoding and Weather

```
SELECT address,  
unnest(  
    geocoder.weather.current(  
        geocoder.usa.geocode_address(address)  
    )  
) FROM places;
```

```
    address = 1023 Lenox Ave, Miami Beach, FL 33139  
    timestamp = 2025-01-31 01:30:00  
    temperature = 21.1  
    wind_speed = 20.8  
wind_direction = 118  
    conditions = Mainly clear
```


Distributed Scalar UDFs

```
{
  "flight_name": "geocoder/usa/uppercase",
  "comment": "upper case a string",
  "input_schema": pa.schema(
    [
      pa.field("input", pa.string()),
    ]
  ),
  # Scalar UDFs have a single field
  "output_schema": pa.schema([
    pa.field("result", pa.string())
  ]),
  "process": uppercase_string,
},
```

Distributed Scalar UDFs

```
def uppercase_string(input: pa.Table) -> list[dict[str, Any]]:  
    return [  
        {"result": row["input"].upper()}  
        for row in input.to_pylist()  
    ]
```

```
SELECT geocoder.usa.uppercase('hello ' || range) as r  
from ids limit 3;  
r = HELLO 0  
r = HELLO 1  
r = HELLO 2
```


What Airport Can Do?

Feature List

Basics	<code>airport_list_flights()</code> <code>airport_take_flights()</code>
Catalog Integration, Schemas and Tables	<code>ATTACH, CREATE SCHEMA, CREATE TABLE</code> <code>INSERT, UPDATE, DELETE, SELECT</code>
Scalar User Defined Functions	<code>SELECT function(value) from source;</code>
Table In/Out Functions	<code>SELECT * from airport_function('Name',</code> <code>(SELECT * from source));</code>
Authentication / Secrets Manager	<code>CREATE SECRET</code>

When will this be ready?

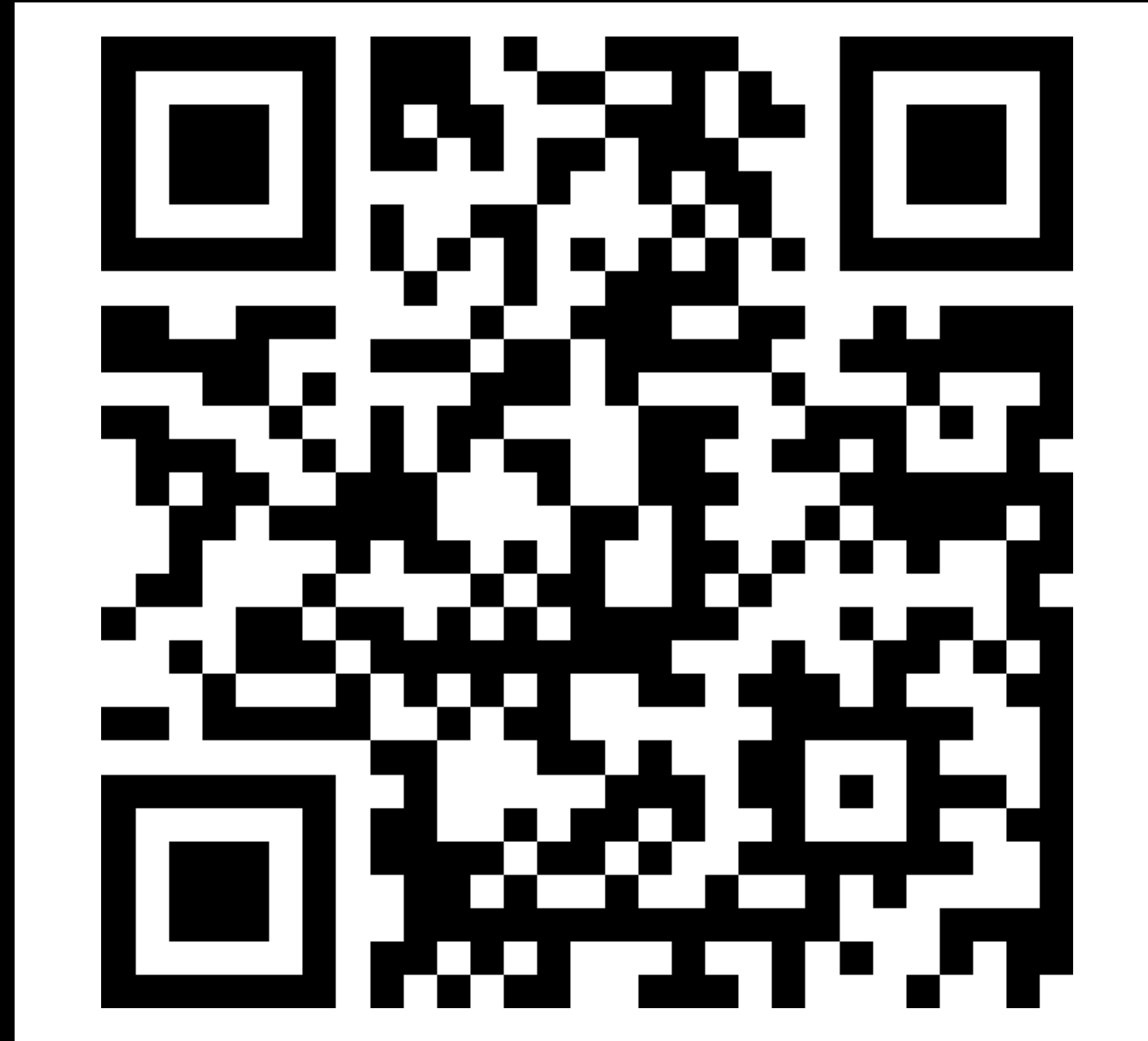
Ship it!

- Will be a community extension. Needs **DuckDB** ≥ 1.2
- MIT Licensed on GitHub.
- <https://github.com/Query-Farm/duckdb-airport-extension>
- Please ★ the repo.
- Send me your questions: rusty@conover.me, follow me on LinkedIn for updates.
- Join the **DuckDB Discord**
- What is “Query Farm”?
A forthcoming collection of Airport Flight Servers for data sources.



Questions and hopefully, answers

Submit your questions here:



<https://app.sli.do/event/tiAGnGKijPD64BgSxHxv2U>