That looks weird!

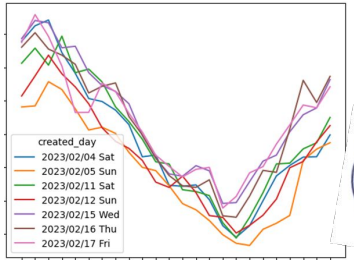
Exploring Mastodon user behaviour with Kafka & DuckDB

data-folks.masto.host/@saubury

@SimonAubury

@saubury

DATA ENG MeetUp



Mastodon user behaviour

What are we talking about today?

Mastodon (1)

What on earth is **Mastodon** and WTF is a decentralized social networking platform?

Approach (2)

Distributed data collection and processing with **Apache Kafka** and **Kafka connect**

Data Analysis (3)

Data analysis with **DuckDB** and **Seaborn**

Demo (4)

Code demo!

(if there's time)

Simon Aubury

Principal Data Engineer

thoughtworks

💗 Kafka enthusiast
🤗 Confluent Community Catalyst
🏠 Sydney, Australia

Mastodon

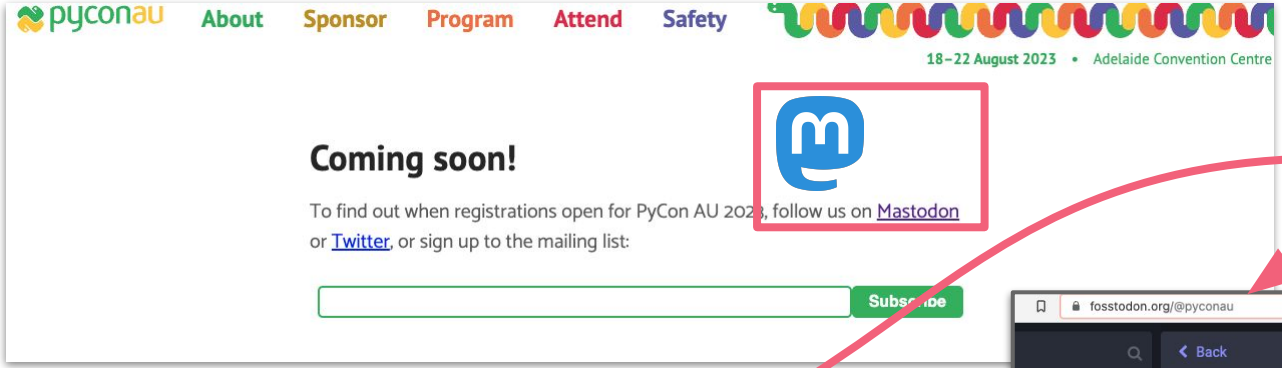
🐘 data-folks.masto.host/@saubury

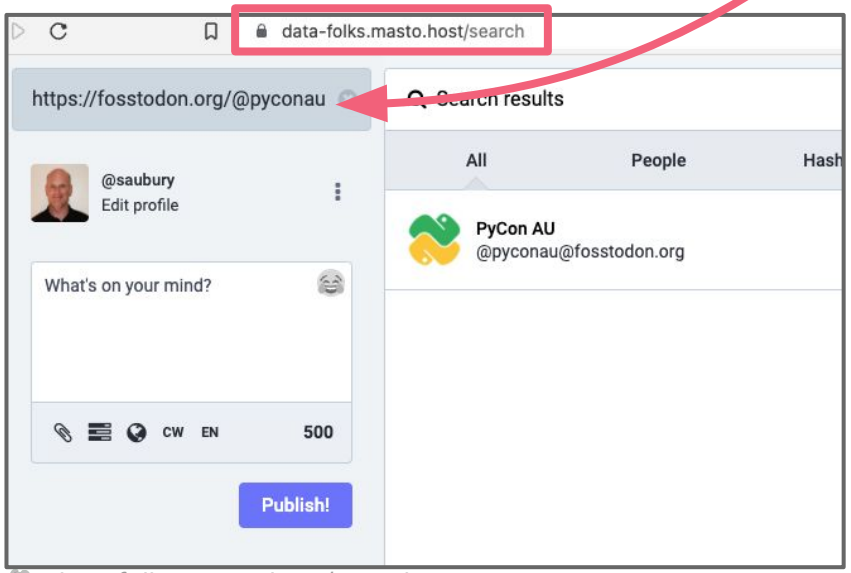
Mastodon

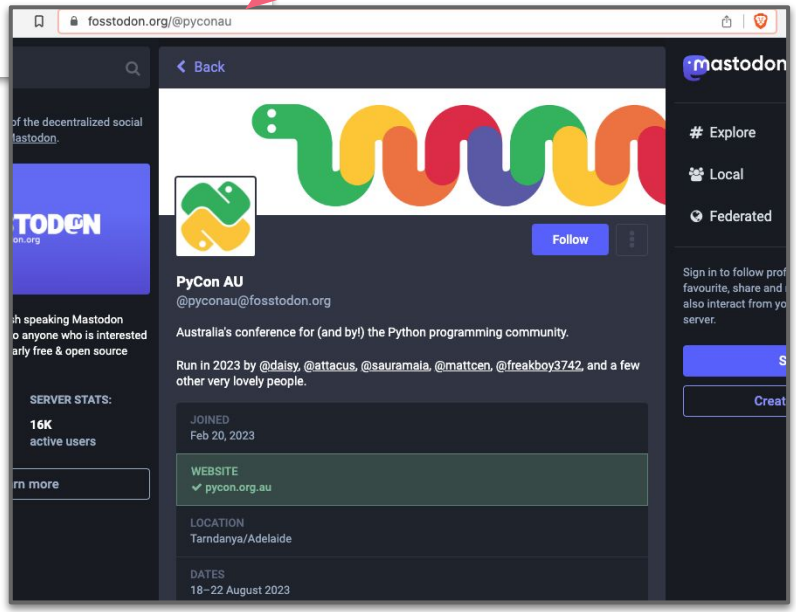
Mastodon is a *decentralized* social networking platform.

- Users are members of a **specific Mastodon instance**
- Servers are capable of joining other servers to **form a federated social network**.

Photo by Battenhall on Unsplash

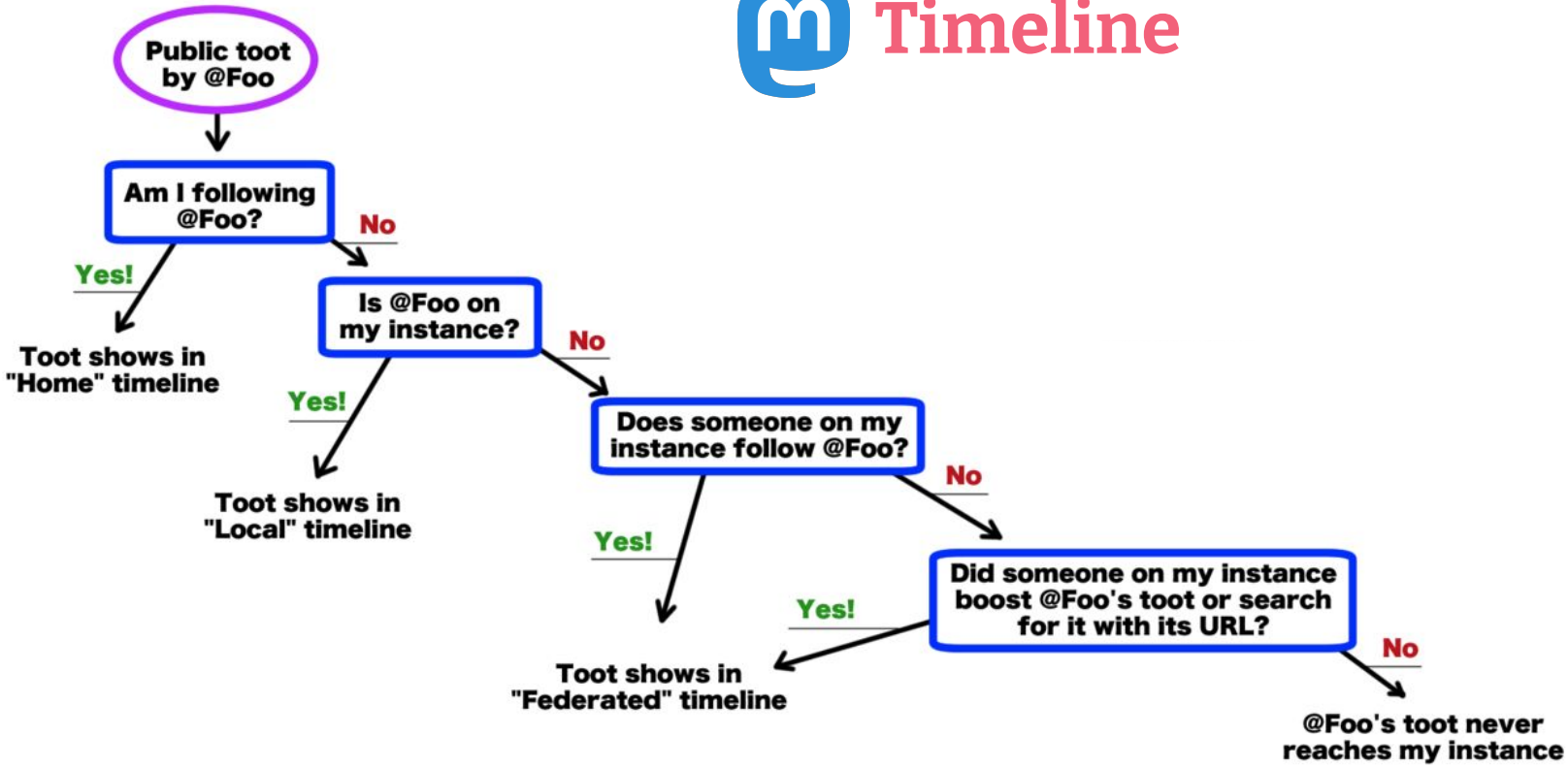






data-folks.masto.host/@saubury

Timeline

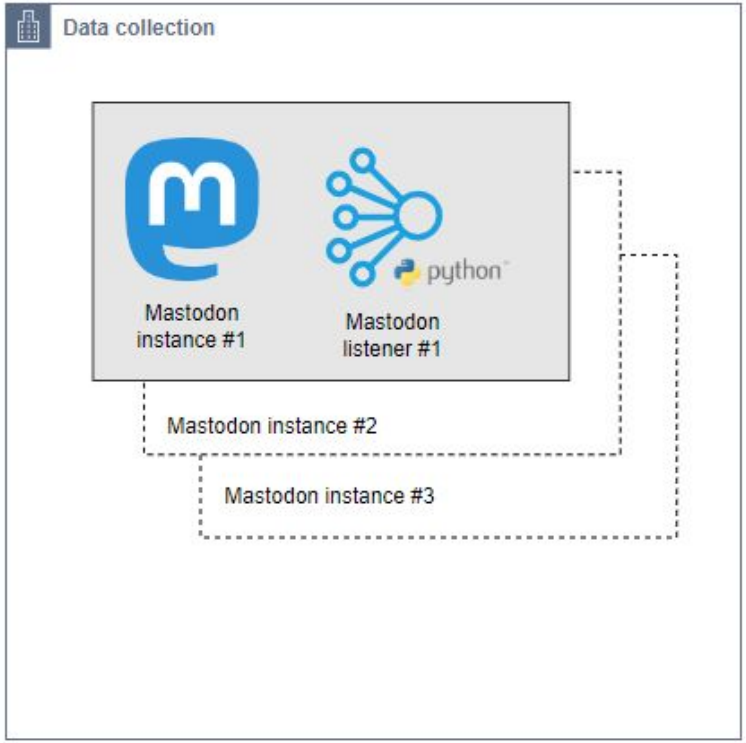


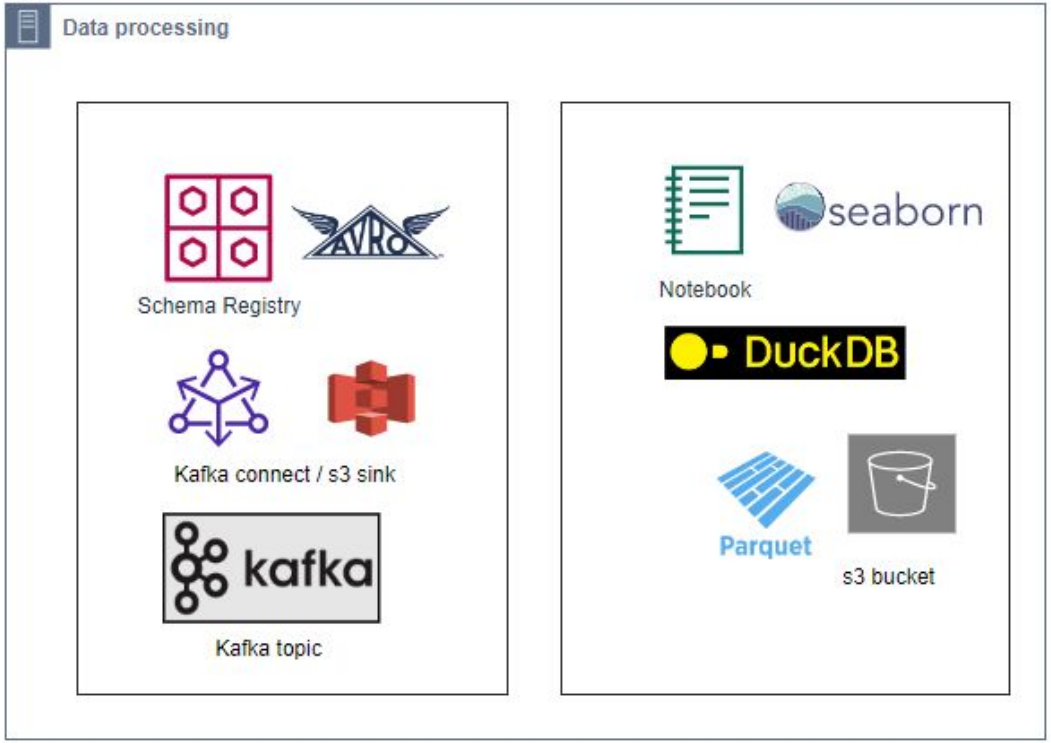
Wikipedia

Data collection

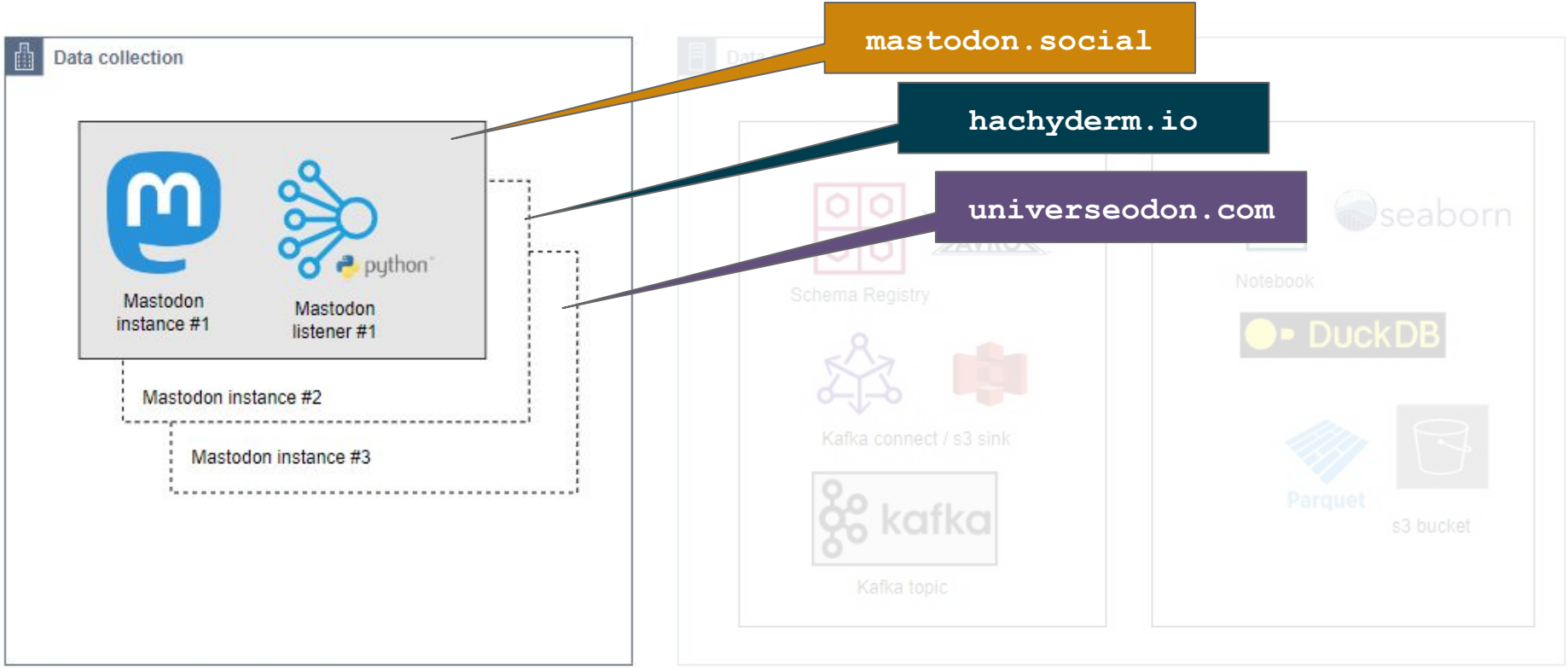
Image by Pexels from Pixabay

Architecture





Data collection



Mastodon Listener

GitHub mastodonlisten.py

Kafka producer

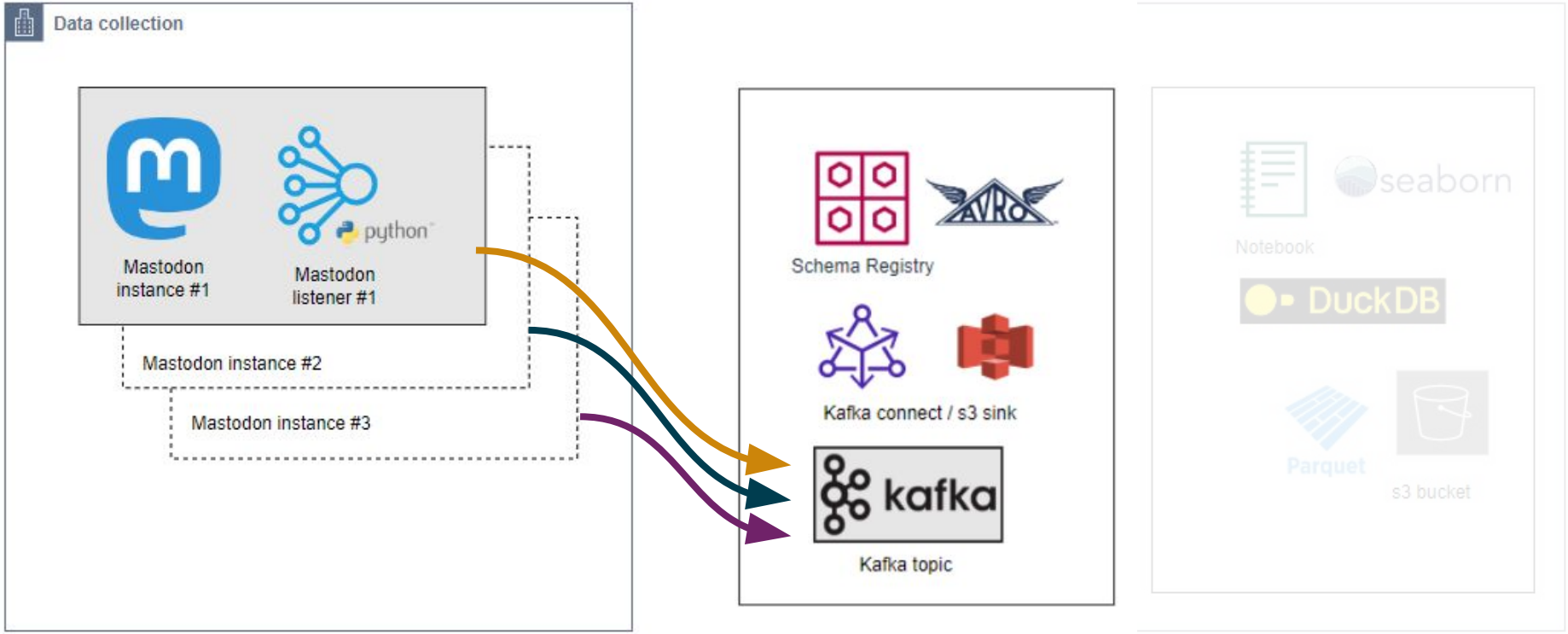
AVRO serializer

Listener

Servers

```python
# Kafka AVRO producer
def kafka_producer():
    producer_config = {
        'bootstrap.servers': 'localhost:9092',
        'schema.registry.url': 'http://localhost:8081',
        'broker.address.family': 'v4'
    }
    value_schema = avro.load('avro/mastodon-topic-value.avsc')
    producer = AvroProducer(producer_config, default_value_schema=value_schema)

# Listener for Mastodon events
class Listener(mastodon.StreamListener):
    def on_update(self, status):
        m_text = BeautifulSoup(status.content, 'html.parser').text
        producer.produce(topic = topic_name, value = value_dict)

    mastodon = Mastodon('https://mastodon.social')
    mastodon.stream_public(Listener())
```

Streaming - Kafka



Authenticating

Mastodon.py

You generally *don't* need your application to be authorized to listen to a mastodon feed.

The docs recommend to **not** authenticate

mastodonpy.readthedocs.io/en/stable/01_general.html

Mastodon.py
stable

Search docs

INTRODUCTION
Mastodon.py
General information
Rate limiting
Pagination
IDs and unpacking
Versioning
A brief note on block lists
Return values
Error handling

API METHODS
App registration, authentication and preferences
Statuses, media and polls
Accounts, relationships and lists
Reading data: Timelines
Instance-wide data and search

advanced one and allows you to just poll in a loop without ever sleeping at all yourself. It is for applications that would rather just pretend there is no such thing as a rate limit and are fine with sometimes not being very interactive.

In addition to the per-user limit, there is a per-IP limit of 7500 requests per 5 minute time slot, and tighter limits on logins. Mastodon.py does not make any effort to respect these.

If your application requires many hits to endpoints that are available without logging in, do consider using Mastodon.py without authenticating to get the full per-IP limit.

Pagination

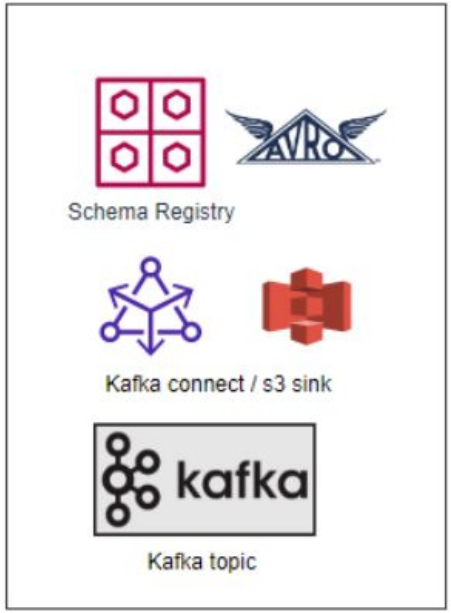
Many of Mastodon's API endpoints are paginated. What this means is that if you request data from them, you might not get all the data at once - instead, you might only get the first few results.

All endpoints that are paginated have four parameters: *since_id*, *max_id*, *min_id* and *limit*. *since_id* allows you to specify the smallest id you want in the returned data, but you will still always get the newest data, so if there are too many statuses between the newest one and *since_id*, some will not be returned. *min_id*, on the other hand, gives you statuses with that minimum id and newer, starting at the given id. *max_id*, similarly, allows you to specify the largest id you want. By specifying either min_id or *max_id* (generally, only one, not both, though specifying both is supported starting with Mastodon

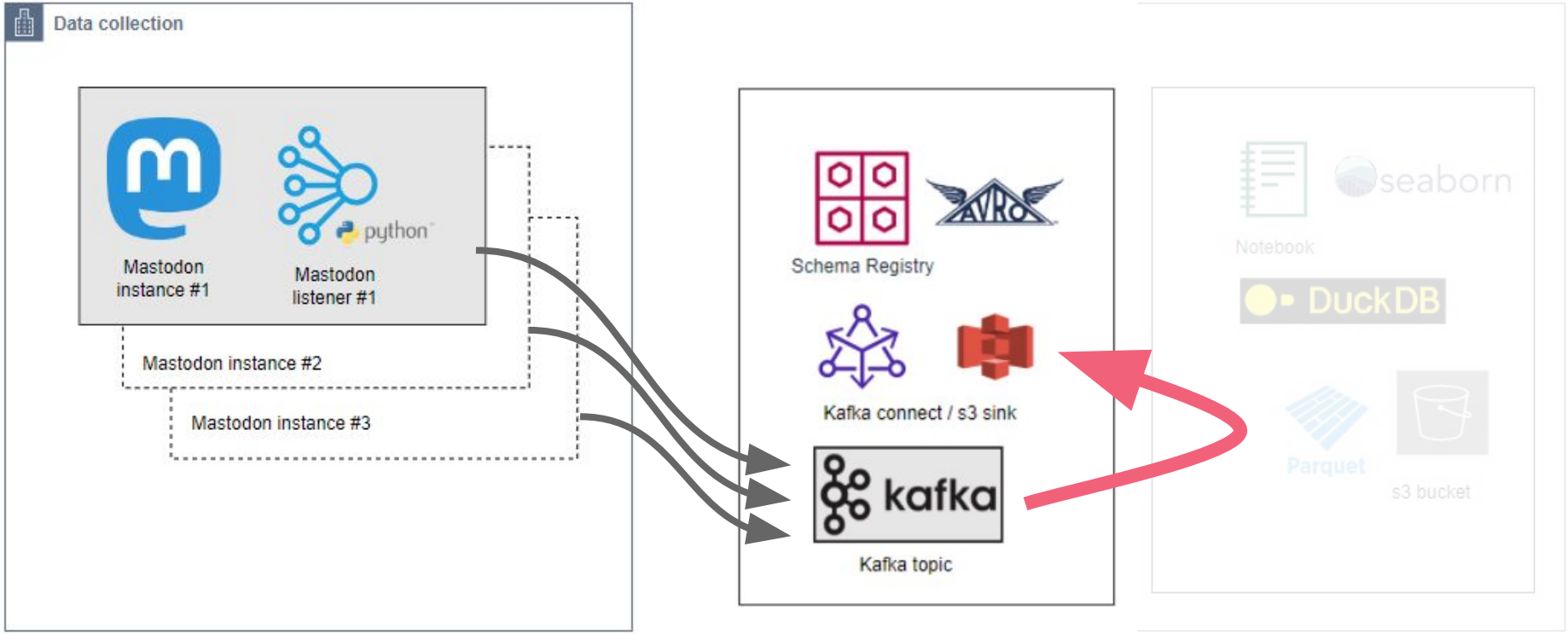
Data storage

Image by Pexels from Pixabay

Data storage



Store to s3


Data collection
Mastodon instance #1
python
Mastodon listener #1
Mastodon instance #2
Mastodon instance #3
Schema Registry
AVRO
Kafka connect / s3 sink
kafka
Kafka topic
seaborn
Notebook
DuckDB
Parquet
s3 bucket

Kafka to s3

Kafka connect / s3 sink

Connector name

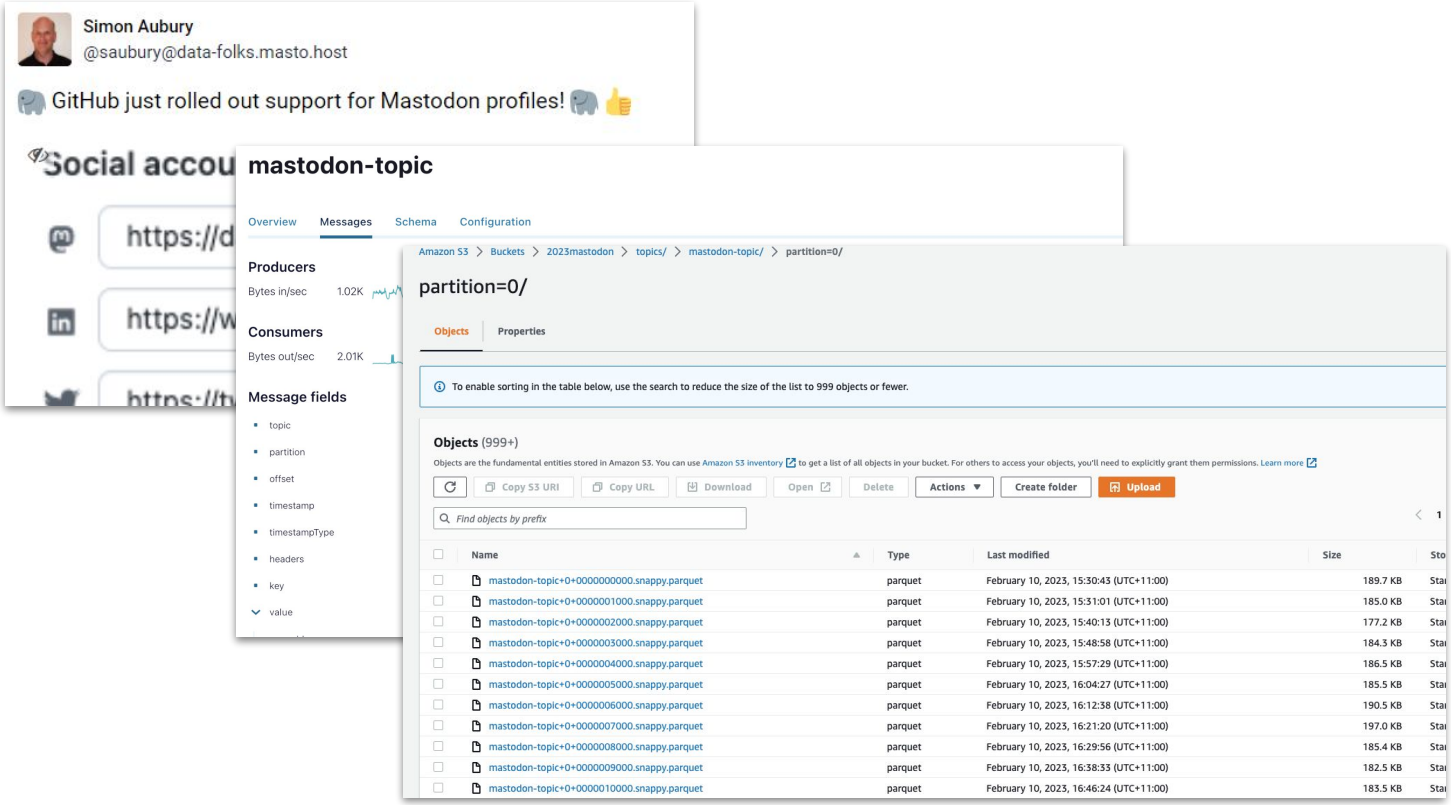
S3 sink class

Kafka topic

Write as parquet

S3 bucket details

```
{
    "name": "mastodon-sink-s3",
    "connector.class": "io.confluent.connect.s3.S3SinkConnector",
    "topics": "mastodon-topic",
    "format.class": "io.confluent.connect.s3.format.parquet.ParquetFormat",
    "flush.size": "10",
    "s3.bucket.name": "mastodon",
    "aws.access.key.id": "minio",
    "aws.secret.access.key": "minio123",
    "storage.class": "io.confluent.connect.s3.storage.S3Storage",
    "store.url": "http://minio:9000"
}
```


Simon Aubury
@saubury@data-folks.masto.host
GitHub just rolled out support for Mastodon profiles!
Social accou
https://d
https://w
mastodon-topic
Overview
Messages
Schema
Configuration
Producers
Bytes in/sec 1.02K
Consumers
Bytes out/sec 2.01K
Message fields
topic
partition
offset
timestamp
timestampType
headers
key
value
Amazon S3 > Buckets > 2023mastodon > topics/ > mastodon-topic/ > partition=0/
partition=0/
Objects
Properties
To enable sorting in the table below, use the search to reduce the size of the list to 999 objects or fewer.
Objects (999+)
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
Name
Type
Last modified
Size
mastodon-topic+0+0000000000.snappy.parquet parquet February 10, 2023, 15:30:43 (UTC+11:00) 189.7 KB
mastodon-topic+0+0000001000.snappy.parquet parquet February 10, 2023, 15:31:01 (UTC+11:00) 185.0 KB
mastodon-topic+0+0000002000.snappy.parquet parquet February 10, 2023, 15:40:13 (UTC+11:00) 177.2 KB
mastodon-topic+0+0000003000.snappy.parquet parquet February 10, 2023, 15:48:58 (UTC+11:00) 184.3 KB
mastodon-topic+0+0000004000.snappy.parquet parquet February 10, 2023, 15:57:29 (UTC+11:00) 186.5 KB
mastodon-topic+0+0000005000.snappy.parquet parquet February 10, 2023, 16:04:27 (UTC+11:00) 185.5 KB
mastodon-topic+0+0000006000.snappy.parquet parquet February 10, 2023, 16:12:38 (UTC+11:00) 190.5 KB
mastodon-topic+0+0000007000.snappy.parquet parquet February 10, 2023, 16:21:20 (UTC+11:00) 197.0 KB
mastodon-topic+0+0000008000.snappy.parquet parquet February 10, 2023, 16:29:56 (UTC+11:00) 185.4 KB
mastodon-topic+0+0000009000.snappy.parquet parquet February 10, 2023, 16:38:33 (UTC+11:00) 182.5 KB
mastodon-topic+0+0000010000.snappy.parquet parquet February 10, 2023, 16:46:24 (UTC+11:00) 183.5 KB

DuckDB

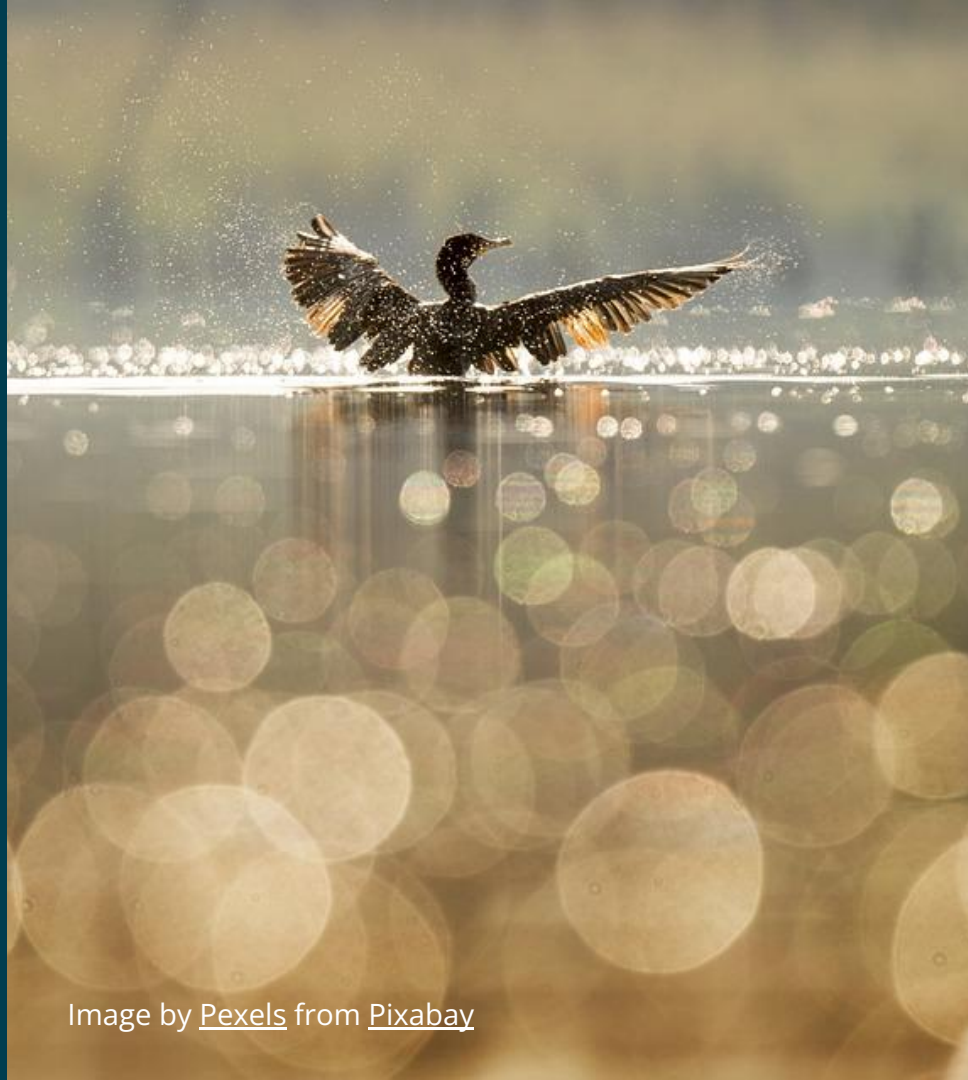
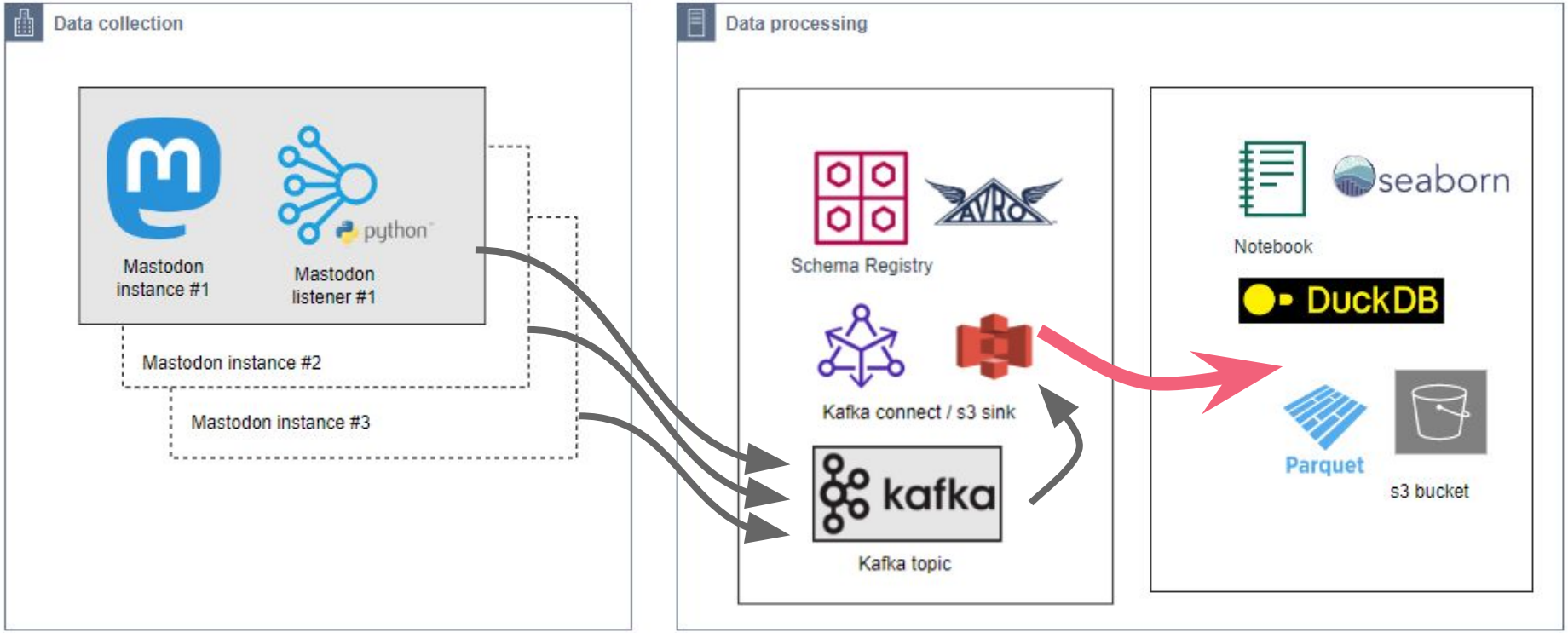


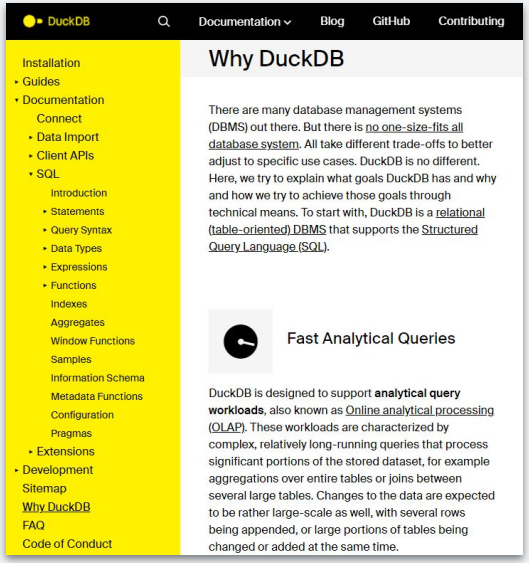
Image by Pexels from Pixabay

Parquet in DuckDB



Why DuckDB

DuckDB is free / open-source



DuckDB Documentation Blog GitHub Contributing

- Installation
- Guides
- Documentation
  - Connect
  - Data Import
  - Client APIs
  - SQL
    - Introduction
    - Statements
    - Query Syntax
    - Data Types
    - Expressions
    - Functions
    - Indexes
    - Aggregates
    - Window Functions
    - Samples
    - Information Schema
    - Metadata Functions
    - Configuration
    - Pragmas
  - Extensions
- Development
- Sitemap
- Why DuckDB
- FAQ
- Code of Conduct

Why DuckDB

There are many database management systems (DBMS) out there. But there is no one-size-fits all database system. All take different trade-offs to better adjust to specific use cases. DuckDB is no different. Here, we try to explain what goals DuckDB has and why and how we try to achieve those goals through technical means. To start with, DuckDB is a relational (table-oriented) DBMS that supports the Structured Query Language (SQL).

Fast Analytical Queries

DuckDB is designed to support **analytical query workloads**, also known as Online analytical processing (OLAP). These workloads are characterized by complex, relatively long-running queries that process significant portions of the stored dataset, for example aggregations over entire tables or joins between several large tables. Changes to the data are expected to be rather large-scale as well, with several rows being appended, or large portions of tables being changed or added at the same time.

https://duckdb.org/why_duckdb.html

RDBMS / SQL / OLAP - Extensive SQL with a large function library, window functions etc. ACID transactional

No external dependencies - compiled into two files, embedded within a host process

Process foreign data without copying, run queries directly on Pandas data, Python and R, APIs for Java, C, C++

Extensions - Httpfs, s3, parquet, json, FTS, geospatial

Installation

https://duckdb.org/docs/installation/index

CLI / macOS

```
brew install duckdb

./duckdb
```

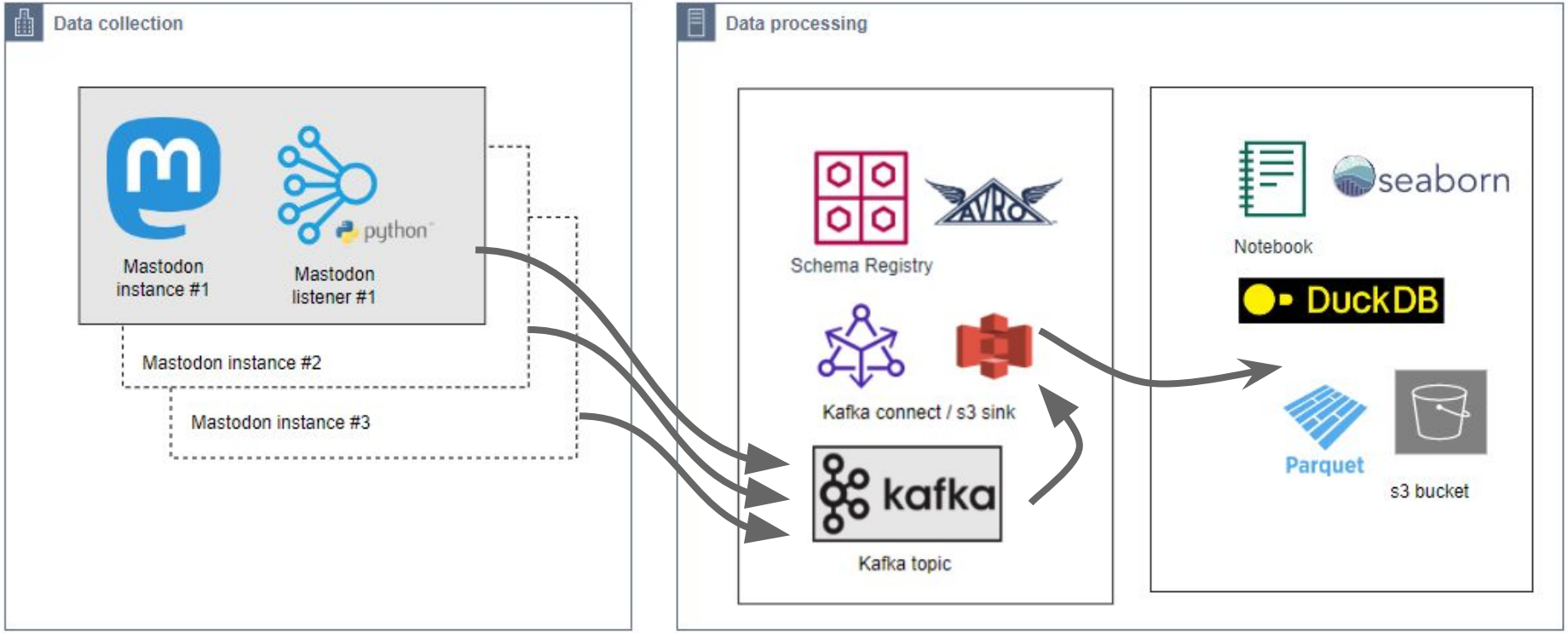
Python

```
pip install duckdb==0.7.1

import duckdb
cursor = duckdb.connect()
print(cursor.execute('SELECT 42').fetchall())
```

node.js

```
npm install duckdb

var duckdb = require('duckdb');
var db = new duckdb.Database(':memory:');
db.all('SELECT 42 AS fortytwo', function(err, res) {
  console.log(res[0].fortytwo)
});
```

Parquet in DuckDB



DuckDB SQL

CLI / macOS

Start DuckDB

```
./duckdb
```

Install extensions

```
INSTALL httpfs;
LOAD httpfs;
```

Set s3 endpoint

```
set s3_access_key_id='XXxxXXxx';
set s3_secret_access_key='XXxxXXxxXXxxXXxxXXxxXXxx';
set s3_region='us-east-1';
```

Select parquet

```
select *
from
read_parquet('s3://mastodon/mastodon-topic/partition=0/*');
```

DuckDB SQL

SQL

1.6 million

```
create table mastodon_toot
as
select m_id, created_at, app, url, username, mastodon_text
from read_parquet('s3://mastodon-topic/partition=0/*');
```

Data Analysis

Photo by Isaac Smith on Unsplash

Daily Mastodon Usage

```
select strftime(created_tz, '%Y/%m/%d %a')
, count(*) as "Num toots"
, count(distinct(username)) as "Num users"
, count(distinct(from_instance)) as "Num urls"
, mode(case when bot='False' then username end)
, mode(case when bot='True' then username end)
, mode(base_url) as "Most freq host"
from mastodon_toot
group by 1 order by 1;
```

SQL / Notebook

Created day	Num toots	Num users	Num urls	Most freq non-bot	Most freq bot	Most freq host
2023/02/03 Fri	17880	8537	1524	gnutiez	nieuws	https://mastodon.social
2023/02/04 Sat	210646	54006	4562	gnutiez	cnexnews	https://mastodon.social
2023/02/05 Sun	191391	49241	4310	IzumiHal	ua	https://mastodon.social
2023/02/06 Mon	41632	17846	2255	gnutiez	nieuws	https://mastodon.social
2023/02/07 Tue	99097	30701	3350	gnutiez	cnexnews	https://mastodon.social
2023/02/08 Wed	188503	49649	4372	gnutiez	cnexnews	https://mastodon.social
2023/02/09 Thu	166096	48532	4227	worldeconomicfella	cnexnews	https://mastodon.social
2023/02/10 Fri	207877	54230	4608	gnutiez	cnexnews	https://mastodon.social

200,000 toots a day from 50,000 users

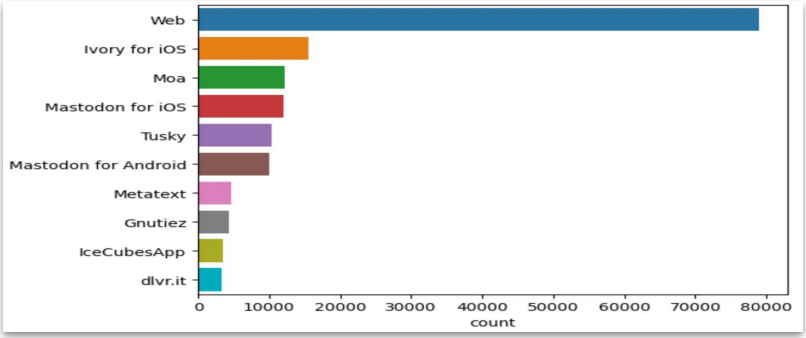
mastodon.social was the most popular host

News organisations are the biggest generator of content

Mastodon App Landscape

SQL / Notebook

```
%%sql
mastodon_app_df <<
    select *
    from mastodon_toot
    where app is not null
    and app <> ''
    and bot='False';

sns.countplot(data=mastodon_app_df, y="app")
```



Mastodon application landscape is rapidly changing

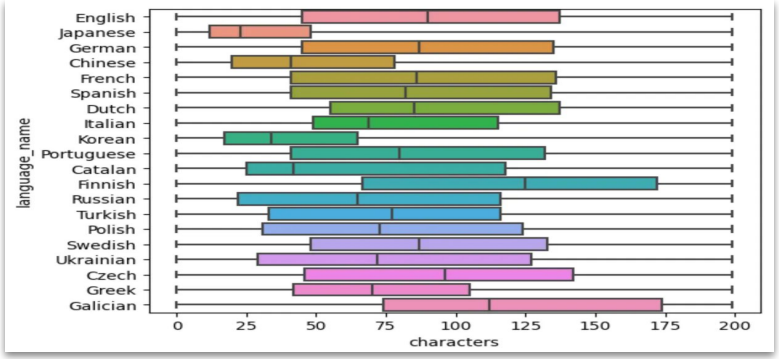
Web usage is the preferred client, with mobile apps like Ivory, Moa, Tusky, and the Mastodon app

Generally, the app attribute does not federate

Toot Length by Language Usage

SQL / Notebook

```
%%sql
mastodon_lang_df <<
    select *
    from mastodon_toot
    where language not in ('unknown');

sns.boxplot(data=mastodon_lang_df,
x="characters", y="language_name", orient="h")
```



Chinese, Japanese, and Korean toots are shorter than English

Galicia and Finish messages are longer

Perhaps logographic languages (like Mandarin) convey more with fewer characters?

Quick statistics

Quick statistics from the data collected over **ten days (February 3 to February 12)**

1,622,149 Mastodon toots seen

142,877 unique Mastodon users

8,309 unique Mastodon instances, **131 languages** seen

Shortest toot is 0 characters, **average toot length is 151** characters, and **longest toot is 68,991** characters

All toots 245,245,677 characters (over 1.6 million toots) in **DuckDB's memory only 745.5MB**

Time it takes to calculate the above statistics in a single SQL query is **0.7 seconds**

Demo

✍️ Medium Blog

🛠️ Code

Slides

Thanks / questions?

Slides

data-folks.masto.host/@saubury

@SimonAubury

Image by David Mark from Pixabay