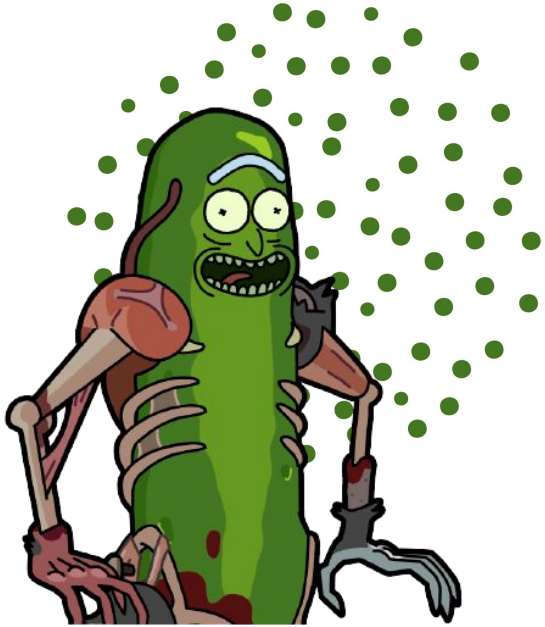


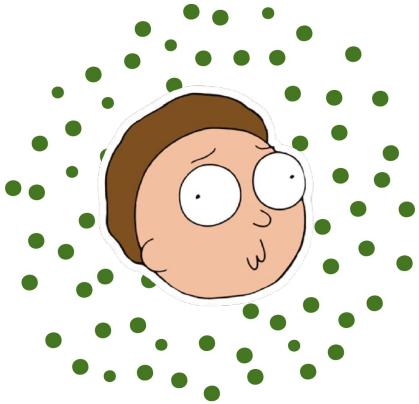
Building an E2E Analytics Pipeline with PyFlink

Marta Paes (@morsapaes)
Developer Advocate

ververica

This Talk

Motivation & Evolution of PyFlink



Demo

Looking ahead

Why Flink + Python?

Java

Scala

Why Flink + Python?

Expose the functionality of Flink beyond the JVM

Why Flink + Python?

Mature and **intuitive** analytics stack...

scikit learn

NumPy

SciPy

pandas

Why Flink + Python?

Mature and **intuitive** analytics stack...

scikit learn

2007

NumPy

1995

SciPy

2001

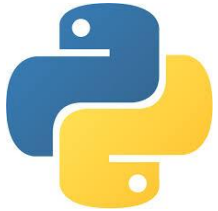
pandas

2008

...that **doesn't really scale**.

Why Flink + Python?

2007



1995

2001

2008

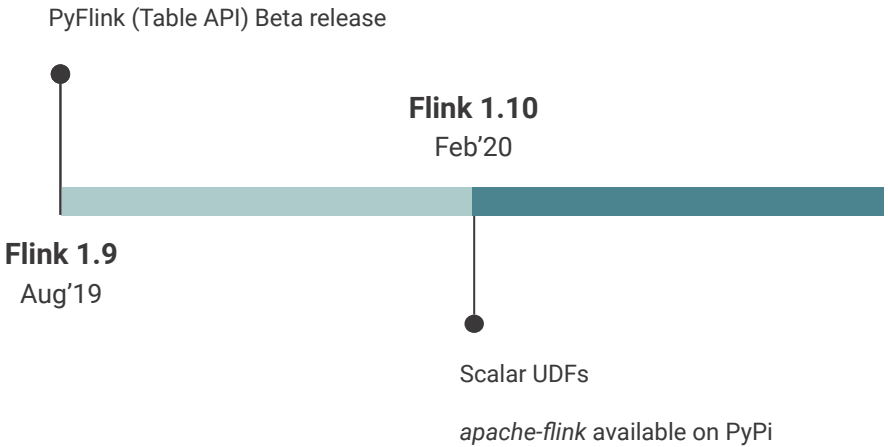
Distribute and **scale** the functionality of Python through Flink

PyFlink Over Time

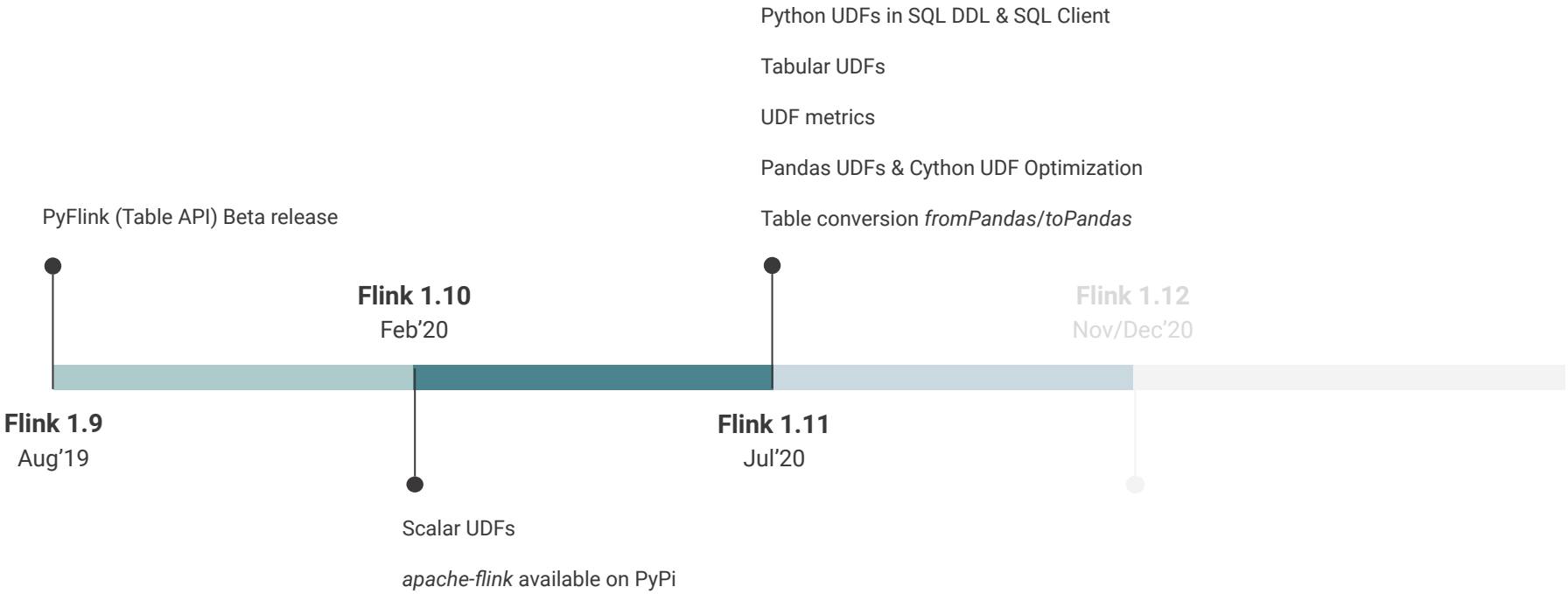
PyFlink (Table API) Beta release

Flink 1.9
Aug'19

PyFlink Over Time

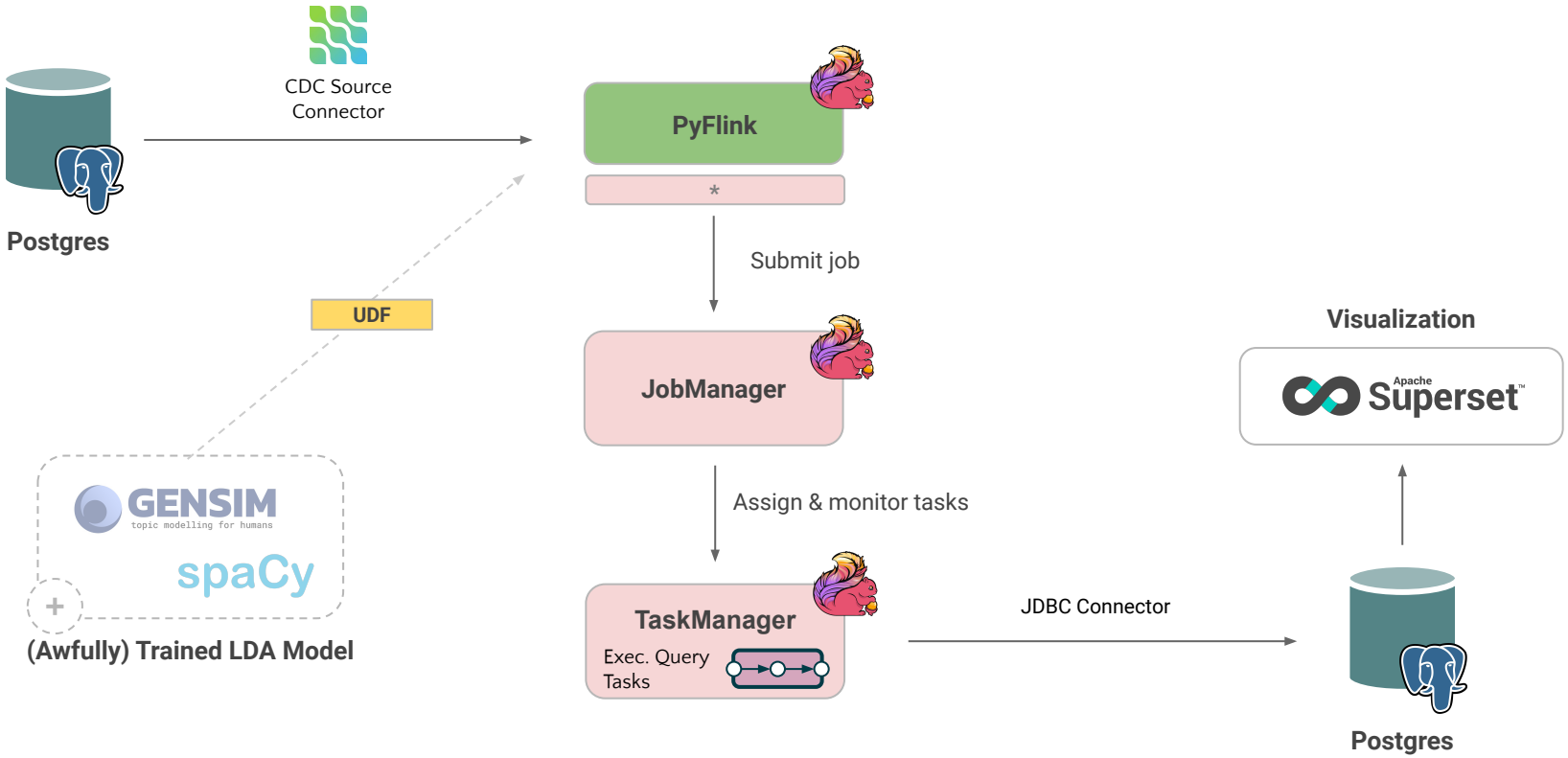


PyFlink Over Time



Can we use PyFlink to identify the most frequent topics in the User Mailing List?

The Demo Environment



DEMO

Step 1. Create the source table.

```
t_sql_source = """CREATE TABLE flink_user_ml (
                        message_date TIMESTAMP(3),
                        message_id STRING,
                        message_in_reply_to STRING,
                        message_from_name STRING,
                        message_from_email STRING,
                        message_subject STRING,
                        message_body_html STRING,
                        message_body_plain STRING,
                        PRIMARY KEY(message_id) NOT ENFORCED
            ) WITH (
                'connector'= 'postgres-cdc',
                'hostname' = 'postgres',
                'username' = 'postgres',
                'password' = 'postgres',
                'database-name' = 'postgres',
                'table-name' = 'stg_flink_user_ml',
                'schema-name' = 'perceval'
)"""

pg_flink_user_ml = t_env.execute_sql(t_sql_source)
```


DEMO

Step 2. Write and register a UDF to clean and classify the messages.

```
@udf(input_types=DataTypes.STRING(), result_type=DataTypes.STRING())
def class_model(m):

    lda = LdaModel.load("model/lda_model/lda_model_user_ml")

    v = tokenizer.pre_process(m)

    vector = lda[v]

    topics = sorted(vector, key=lambda x: x[1], reverse=True)

    return str(topics[0][0])


t_env.register_function("CLASS_TOPIC", class_model)
```


DEMO

Step 3. Build your query and create a sink table to where you'll be inserting the results!

```
topics = t_env.from_path("flink_user_ml") \
              .filter("message_subject.isNotNull") \
              .select("message_id, message_date, message_from_name, CLASS_TOPIC(message_subject)")

topics.execute_insert("flink_user_ml_topics")
```

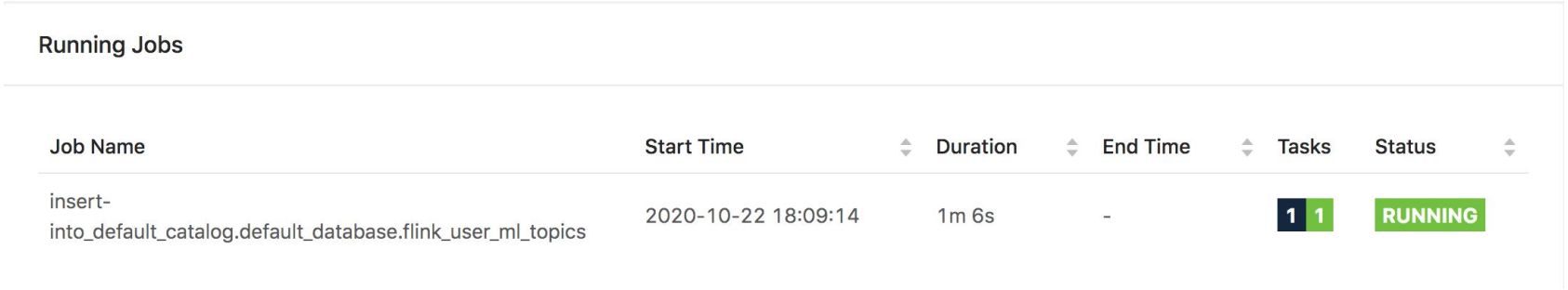
```
t_sql_sink = """CREATE TABLE flink_user_ml_topics (
                        message_id STRING,
                        message_date TIMESTAMP(3),
                        message_from_name STRING,
                        topic VARCHAR(2),
                        PRIMARY KEY (message_id) NOT ENFORCED
              ) WITH (
                   'connector'= 'jdbc',
                   'url' = 'jdbc:postgresql://postgres:5432/postgres',
                   'table-name' = 'perceval.stg_flink_user_ml_topics',
                   'username' = 'postgres',
                   'password' = 'postgres'
)"""

pg_flink_user_ml_topics = t_env.execute_sql(t_sql_sink)
```


DEMO

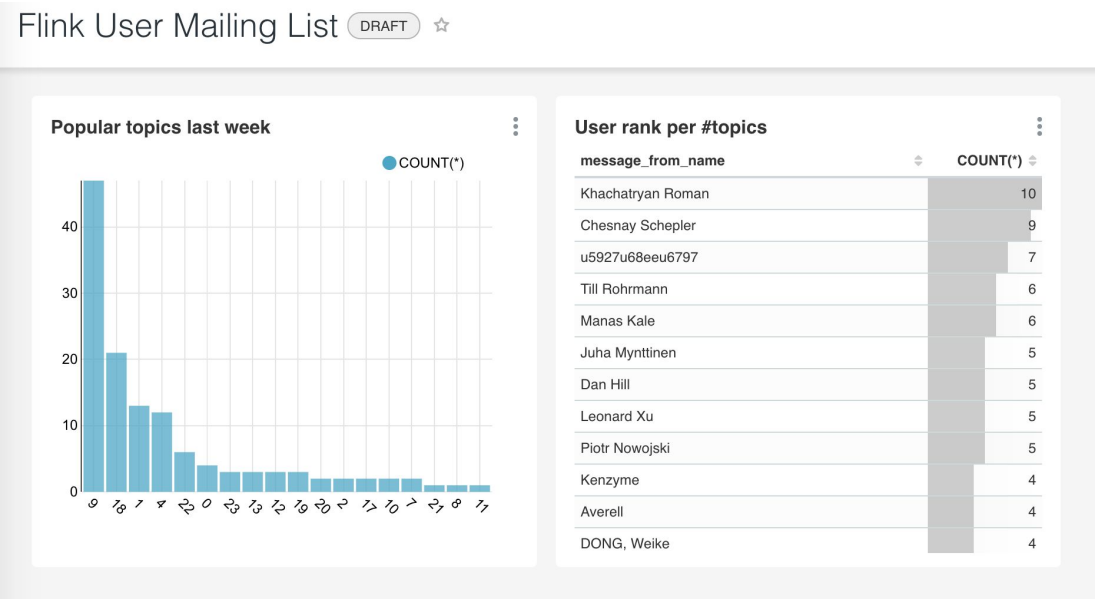
Step 4. Submit the job (and dependencies) to the cluster

```
docker-compose exec jobmanager ./bin/flink run -py /opt/pyflink-ff2020/pipeline.py \
--pyArchives /opt/pyflink-ff2020/lda_model.zip#model \
--pyFiles /opt/pyflink-ff2020/tokenizer.py -d
```



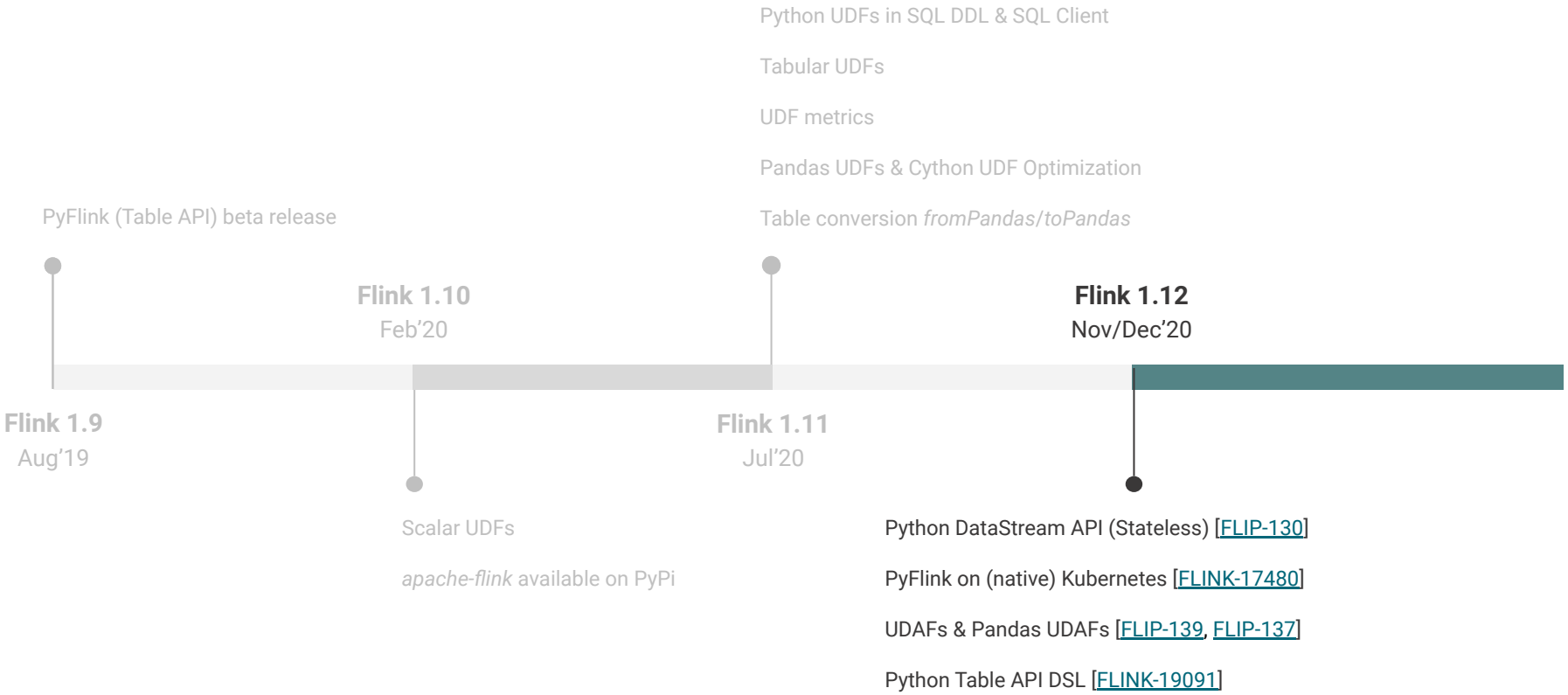
DEMO

Step 5. Visualize in Superset!



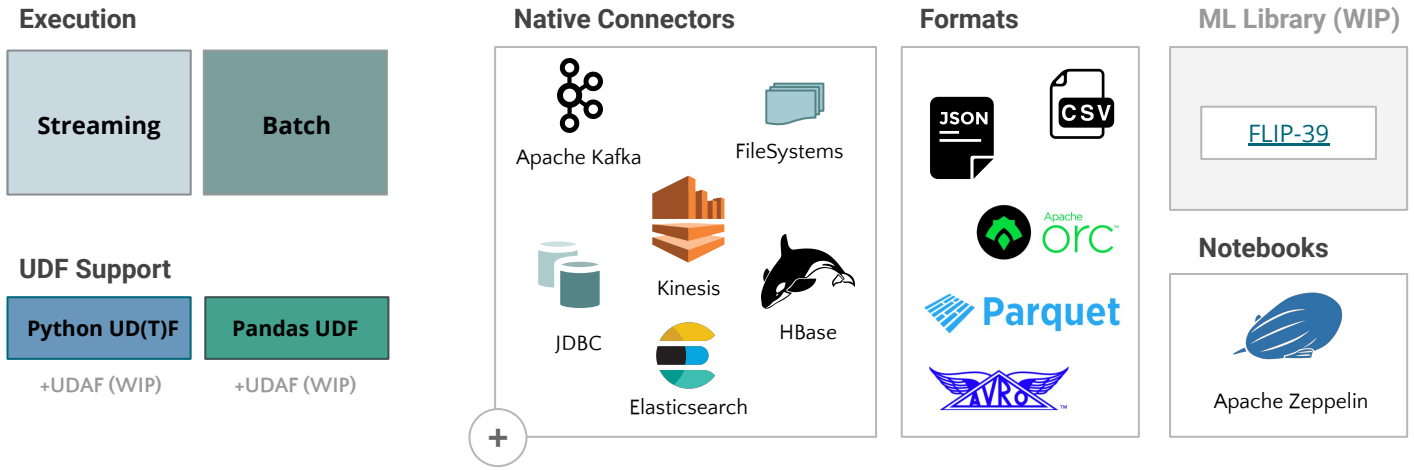

```
(9, '0.521*"flink" + 0.065*"issue" + 0.049*"source" + 0.041*"yarn" + 0.033*"build" + 0.028*"latency"
+ 0.027*"access" + 0.023*"progress" + 0.016*"run" + 0.015*"org"')
```

And soon!

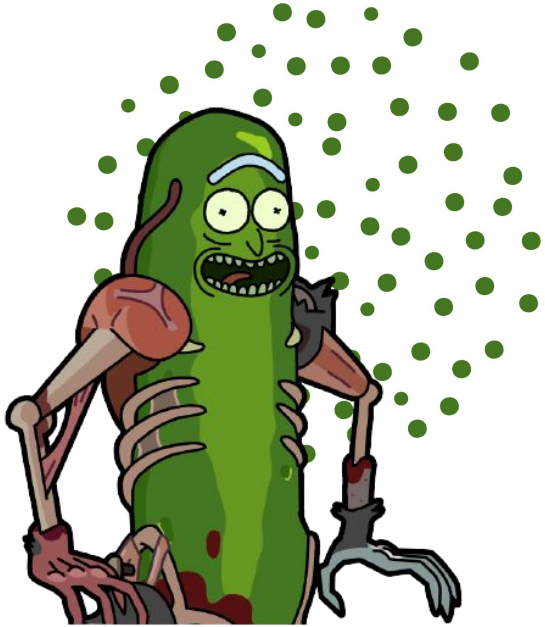

Python UDFs in SQL DDL & SQL Client
Tabular UDFs
UDF metrics
Pandas UDFs & Cython UDF Optimization
Table conversion fromPandas/toPandas
PyFlink (Table API) beta release
Flink 1.10
Feb'20
Flink 1.12
Nov/Dec'20
Flink 1.9
Aug'19
Flink 1.11
Jul'20
Scalar UDFs
apache-flink available on PyPi
Python DataStream API (Stateless) [FLIP-130]
PyFlink on (native) Kubernetes [FLINK-17480]
UDAFs & Pandas UDAFs [FLIP-139, FLIP-137]
Python Table API DSL [FLINK-19091]

PyFlink in a Nutshell

- Unified APIs for batch and streaming
- Native SQL integration + Pandas integration
- Support for a large set of operations (incl. complex joins, windowing, pattern matching/CEP)



Try the PyFlink playground: https://github.com/apache/flink-playgrounds



Thank you, Flink Forward!

Follow me on Twitter: @morsapaes

Learn more about Flink: https://flink.apache.org/

ververica