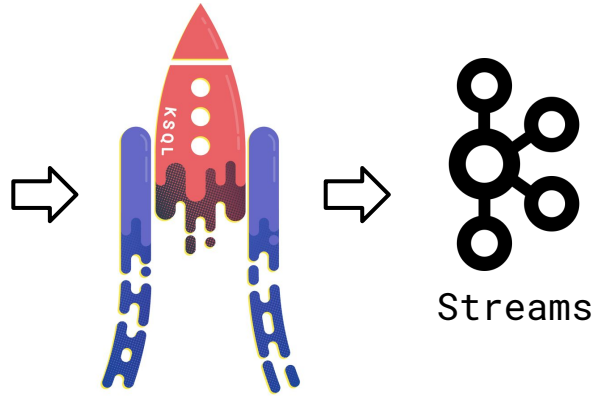
KSQL

Stream Processing with KSQL

Introduction to KSQL

KSQL transforms SQL queries into Kafka Streams applications

```
CREATE TABLE user(
    username VARCHAR,
    email VARCHAR
) WITH (
    KAFKA_TOPIC='user_topic',
    VALUE_FORMAT='json'
);
```



KSQL Architecture

- Open-sourced in 2018 by Confluent
- Built in Java, Scala and runs on the JVM
- KSQL is built on top of the Java Kafka Streams Library
- Kafka as Changelog, RocksDB for local storage
- Can run interactive and on-demand queries
- Supports REST API, CLI, and File-based querying

Comparing KSQL

- Simpler to write SQL than to build entire custom app
- Great for exploration when exact solution not known
- No need for specific programming language skills
- Comes with logging and metrics out of the box
- Not always easy to use SQL to solve all problems
- Can’t import whatever library you want; Supports UDFs

Turning Topics into Tables and Streams

The KSQL CLI can show what topics are available

```
ksql> SHOW TOPICS;

 Kafka Topic     | Registered | Partitions | Partition Replicas | Consumers | ConsumerGroups
-------------------------------------------------------------------------------------------
 purchases       | false      | 1          | 1                  | 0         | 0
 users           | false      | 1          | 1                  | 0         | 0
 connect-configs | false      | 1          | 1                  | 0         | 0
 connect-offsets | false      | 25         | 1                  | 0         | 0
 connect-status  | false      | 5          | 1                  | 0         | 0
-------------------------------------------------------------------------------------------
```

Creating a Stream

Streams can be created from a Kafka Topic or from a query

```
CREATE STREAM purchases (
    username VARCHAR,
    currency VARCHAR,
    amount INT
);

CREATE STREAM purchases_high_value AS
  SELECT *
  FROM purchases"purchases",
  WHERE amount > 100000;
);
```

Demo: Creating a Stream

Creating a Table

Tables can be created from a Kafka Topic or from a query

```
CREATE TABLE users (
    username VARCHAR,
    address VARCHAR,
    email VARCHAR,
    phone_number VARCHAR
) WITH (
    KAFKA_TOPIC='purchases',
    VALUE_FORMAT='JSON',
    KEY='username'
)
```

```
CREATE TABLE high_value AS
  SELECT *
  FROM purchases
  WHERE amount > 100000;
```

Demo: Creating a Table

Querying Syntax

KSQL supports familiar SQL query syntax such as SELECT … WHERE …

```
SELECT username
FROM purchases
WHERE currency="USD" AND
  amount > 100000
```

Demo: Querying

Windowing

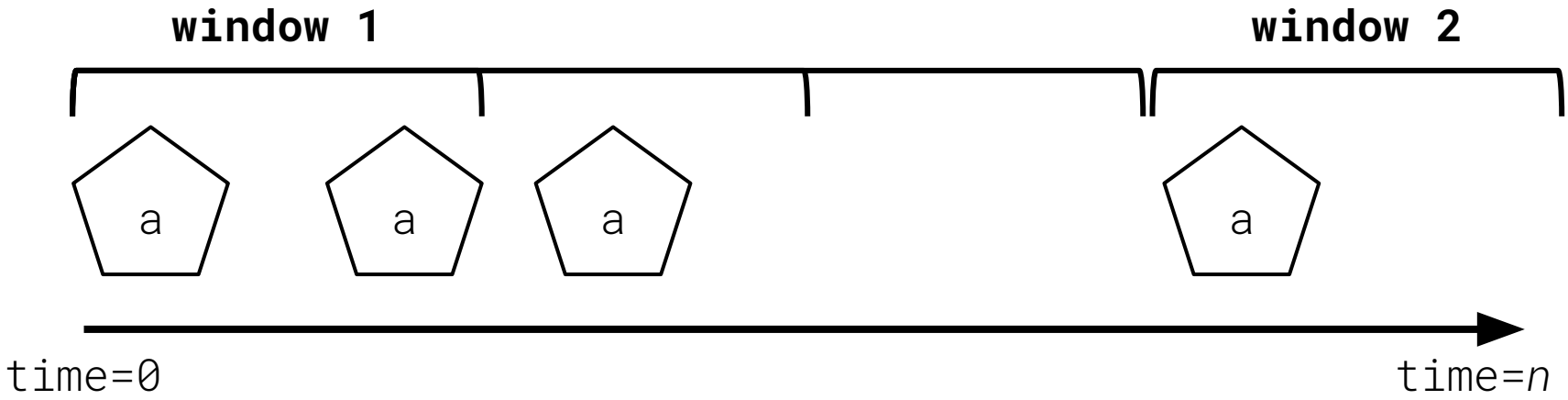
Hopping and Tumbling Windowing

KSQL Supports Tumbling and Hopping Windows on Tables

```
SELECT currency, SUM(amount)
FROM purchase
WINDOW HOPPING (SIZE 10 MINUTES, ADVANCE BY 1 MINUTES)
GROUP BY currency;
```

Session Windowing

Session windows extend indefinitely as long as activity is repeated within the window timeframe



Session Windowing

Session windows extend indefinitely as long as activity is repeated within the window timeframe

```
SELECT currency, SUM(amount)
FROM purchase
WINDOW SESSION (30 MINUTES)
GROUP BY currency;
```

Demo: KSQL Windowing

Aggregating Data

`GROUP BY` allows re-partitioning of a stream or table on a new key

```
SELECT currency, SUM(amount)
FROM purchase
WINDOW SESSION (30 MINUTES)
GROUP BY currency;
```

Aggregating Data

`GROUP BY` allows re-partitioning of a stream or table on a new key

```
SELECT currency, HISTOGRAM(currency)
FROM purchase
WINDOW SESSION (30 MINUTES)
GROUP BY currency;
```

Aggregating Data

`GROUP BY` allows re-partitioning of a stream or table on a new key

```
SELECT currency, TOPK(amount, 10)
FROM purchase
WINDOW TUMBLING (SIZE 1 HOURS)
GROUP BY currency;
```

Demo: Aggregating Data

Joins

Joins Overview

KSQL supports `JOIN` operations on co-partitioned streams and/or tables

```
SELECT p.username, p.amount, u.email
FROM purchase
JOIN users u ON p.username = u.username
```

Demo: Hands-on with Joins

Summary

Summary