confluent

Architecting Microservices Applications with Instant Analytics

Tim Berglund, Sr. Director, Developer Experience, Confluent
Rachel Pedreschi, Sr.l Director, Global Field Engineering, Imply Data

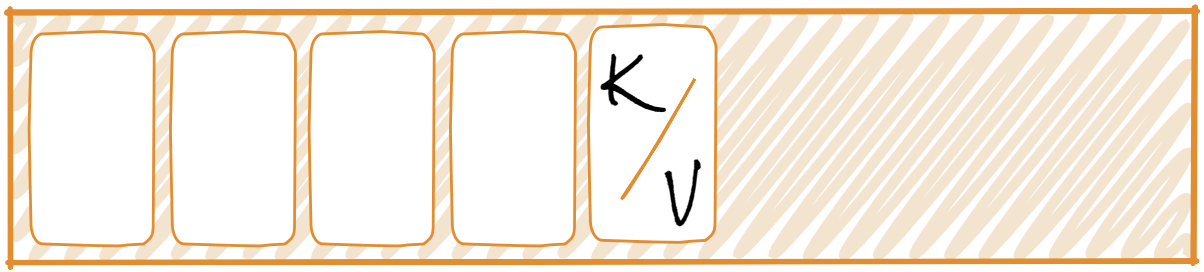
What the heck is Apache Kafka and Why Should I Care?

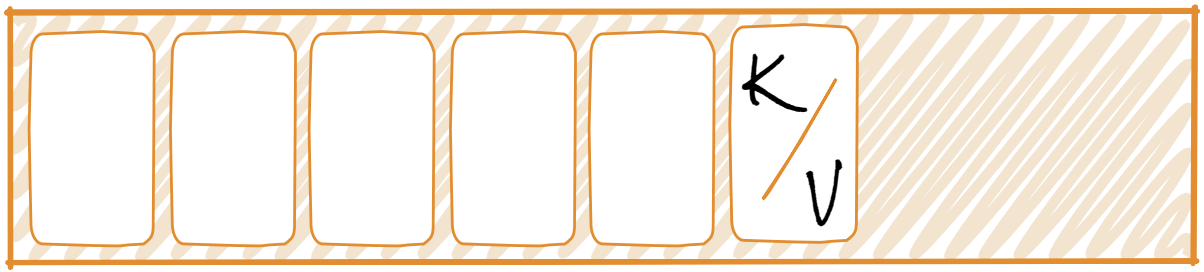

K
V

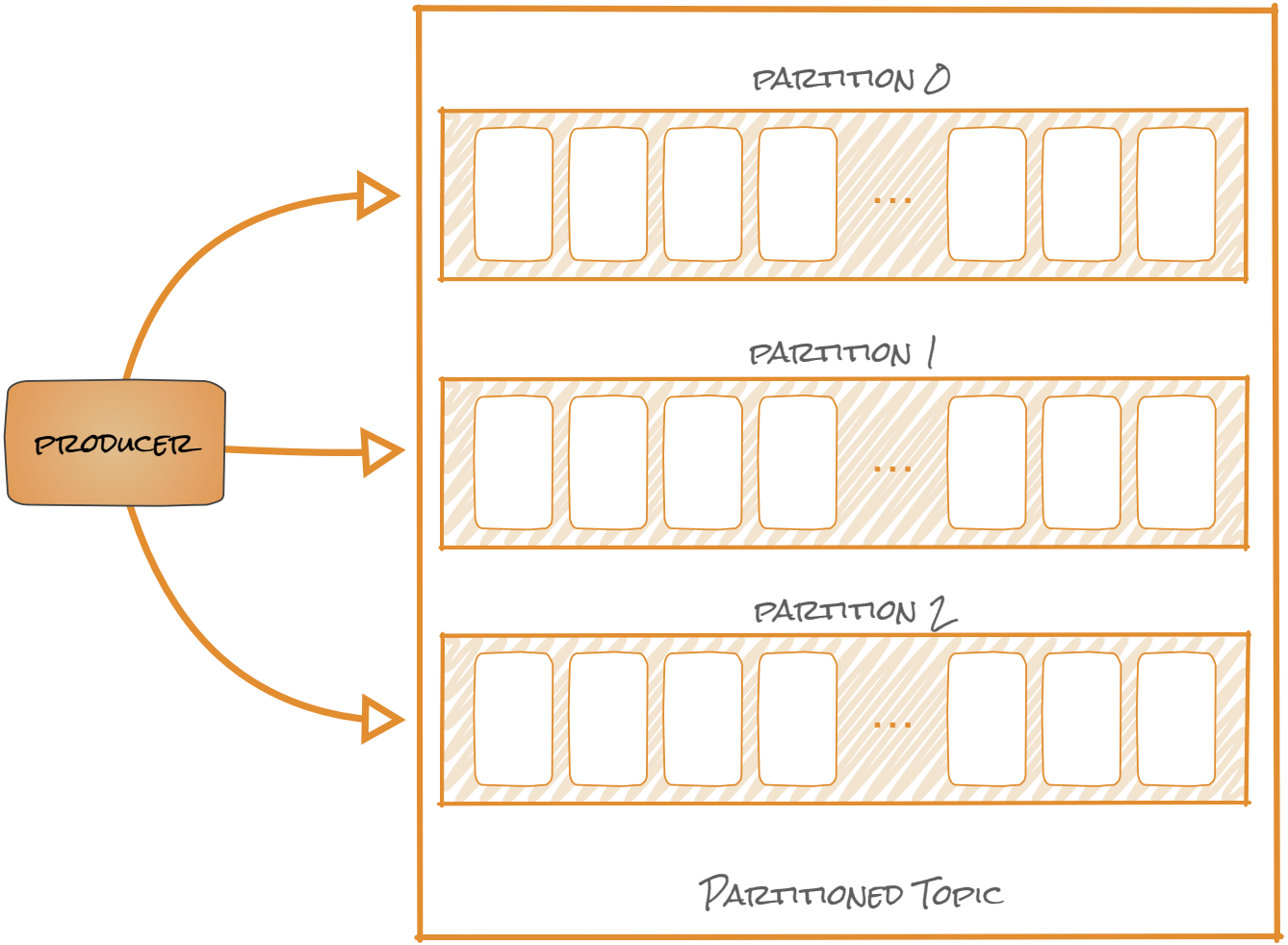

K
V

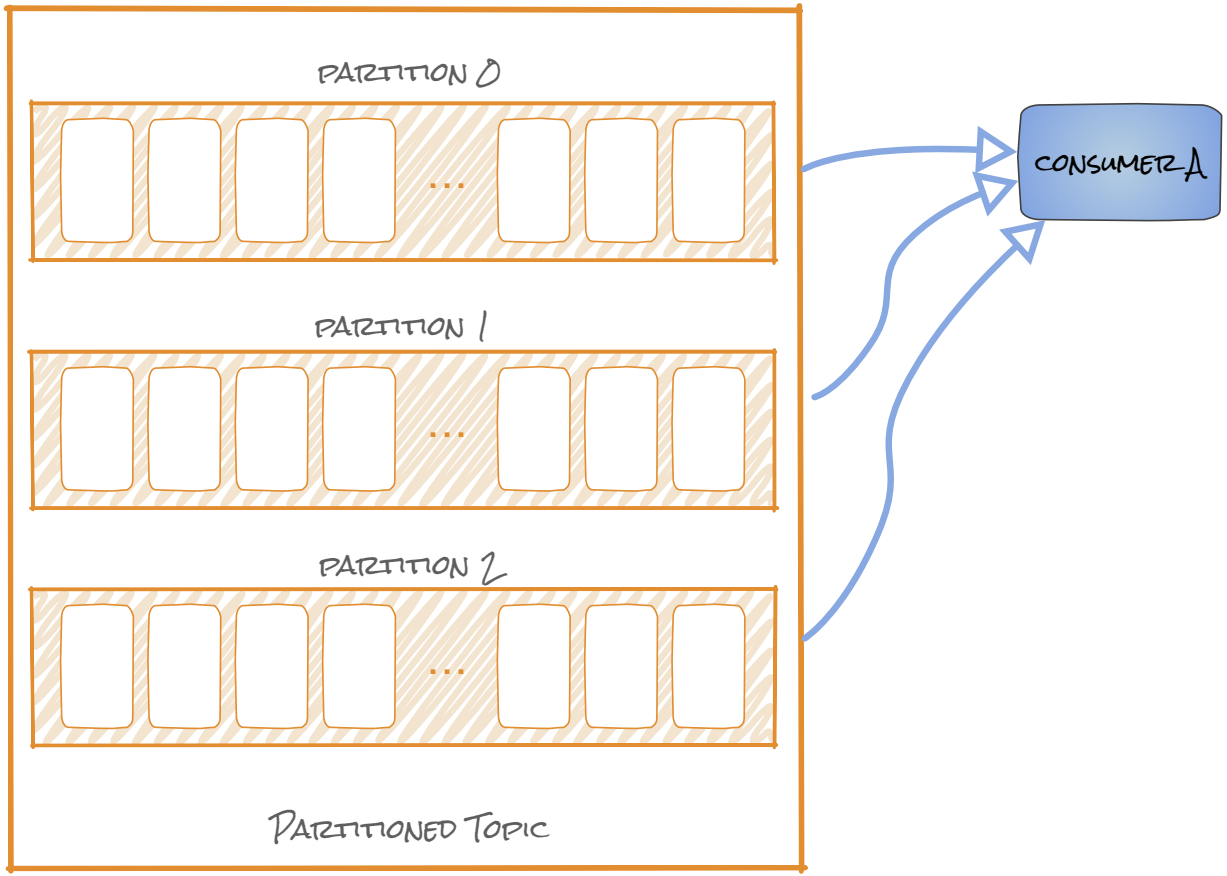

K
V

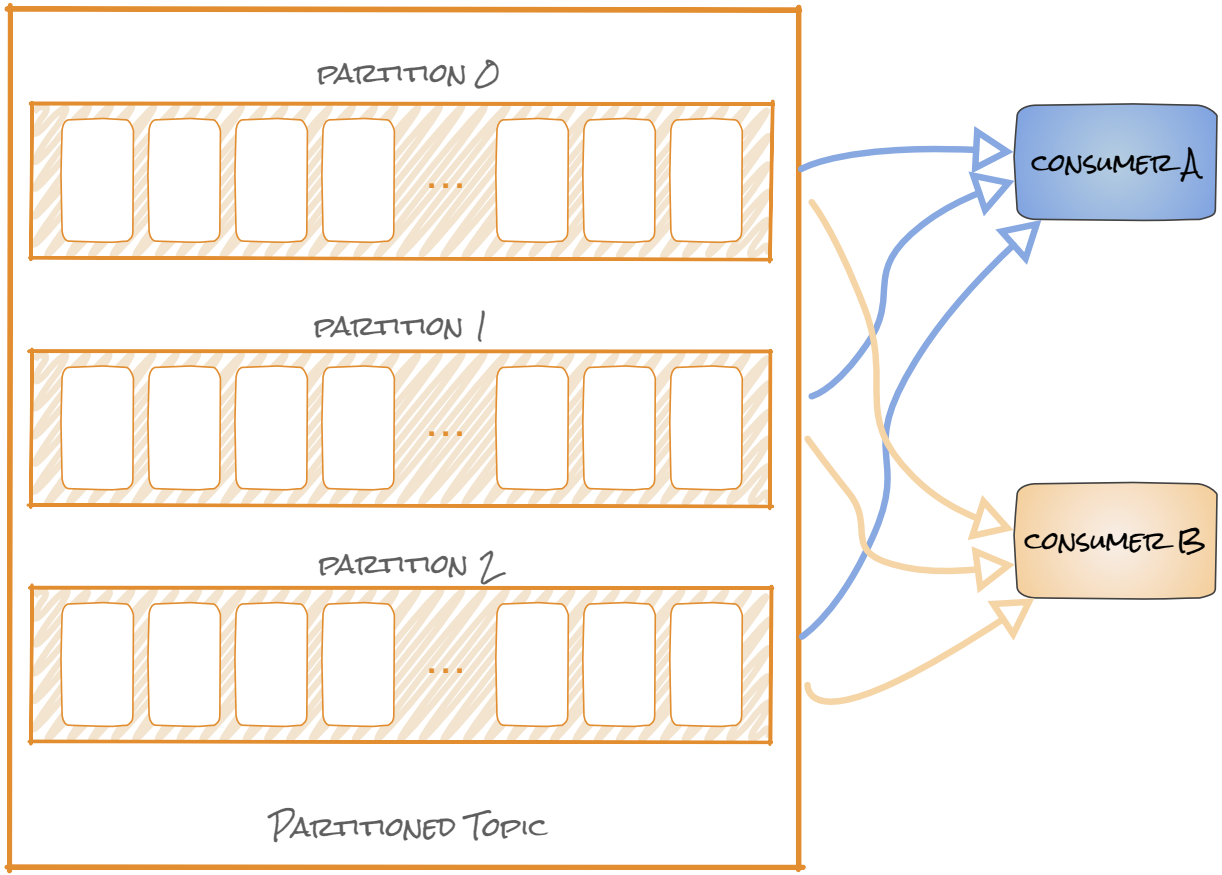

K
V

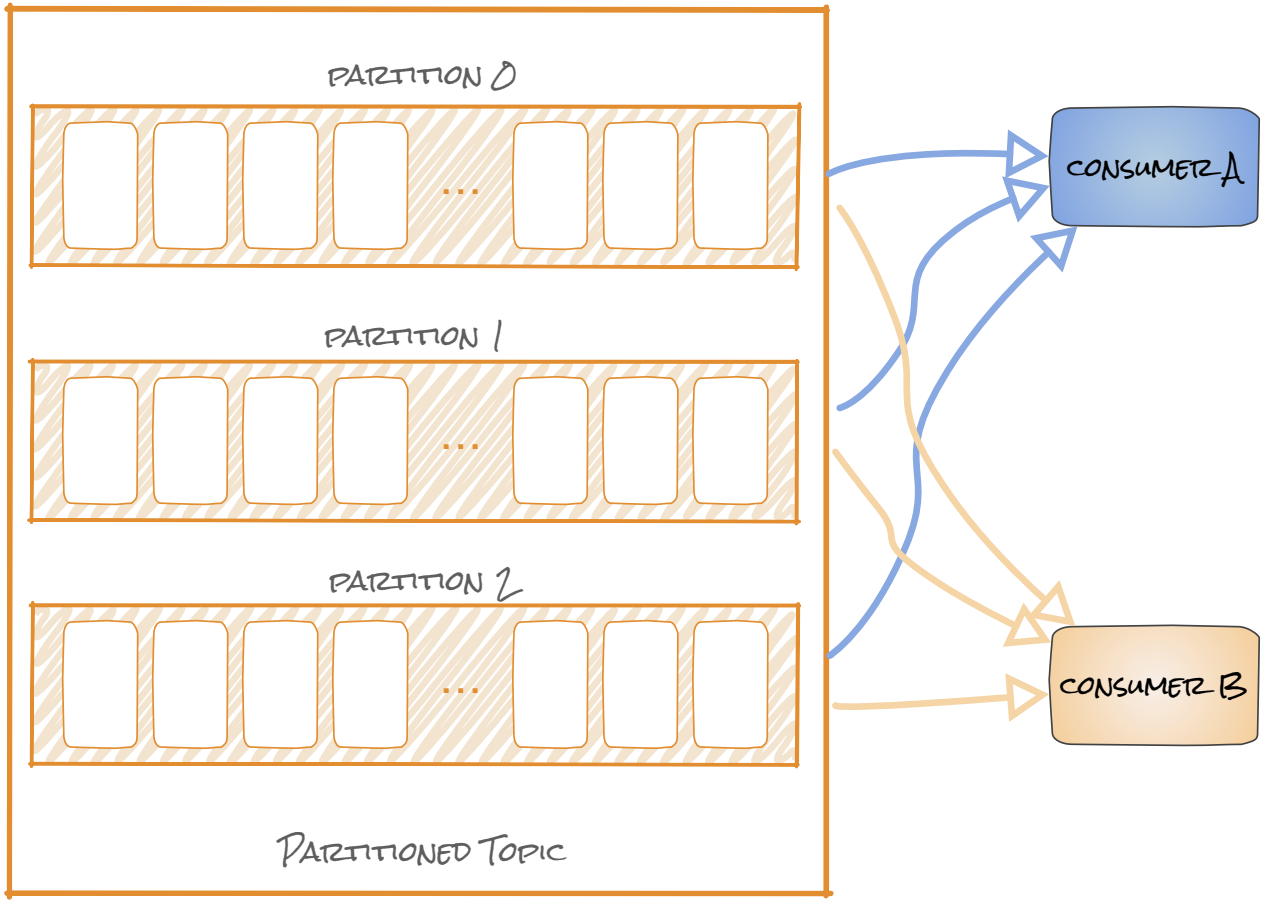

K
V

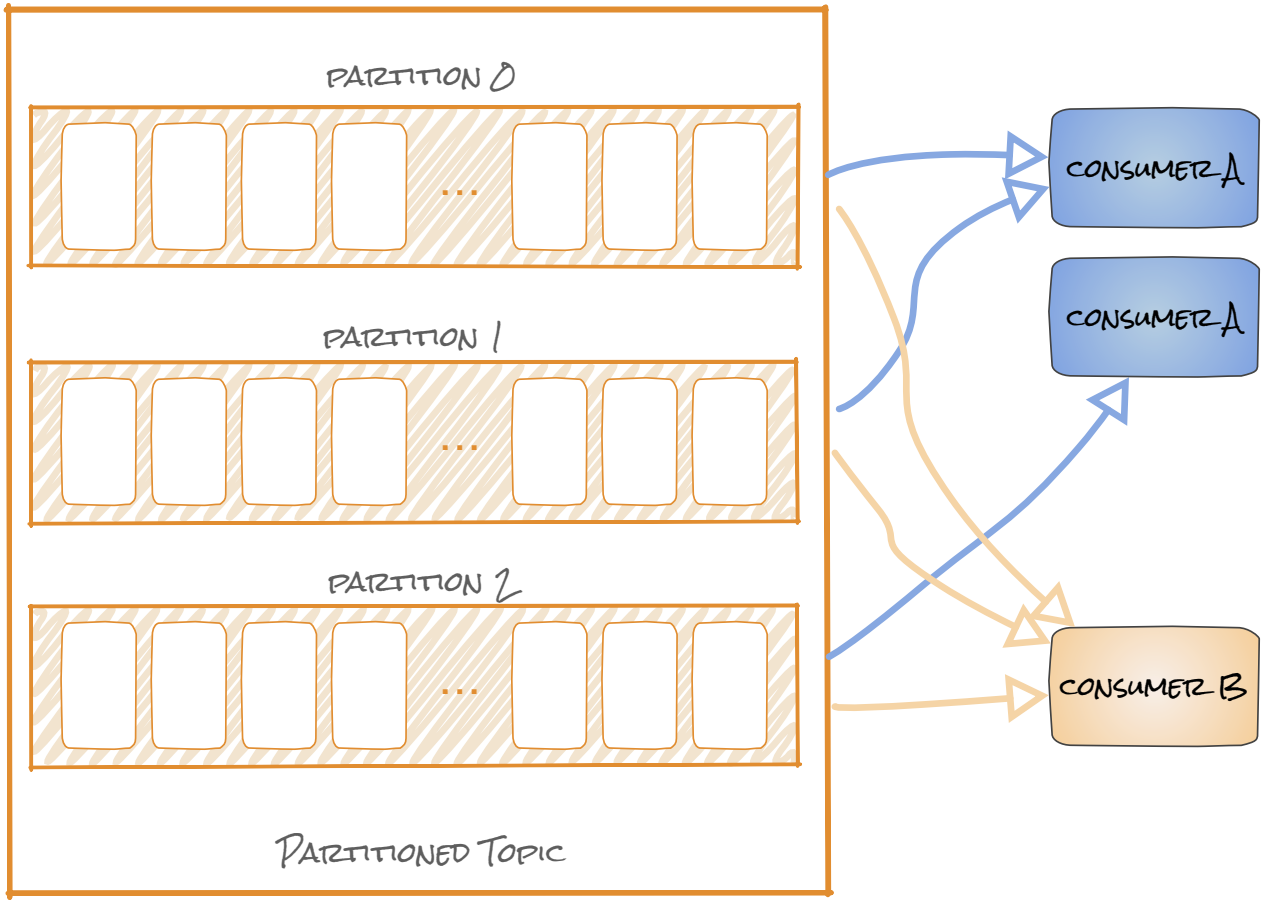

K
V

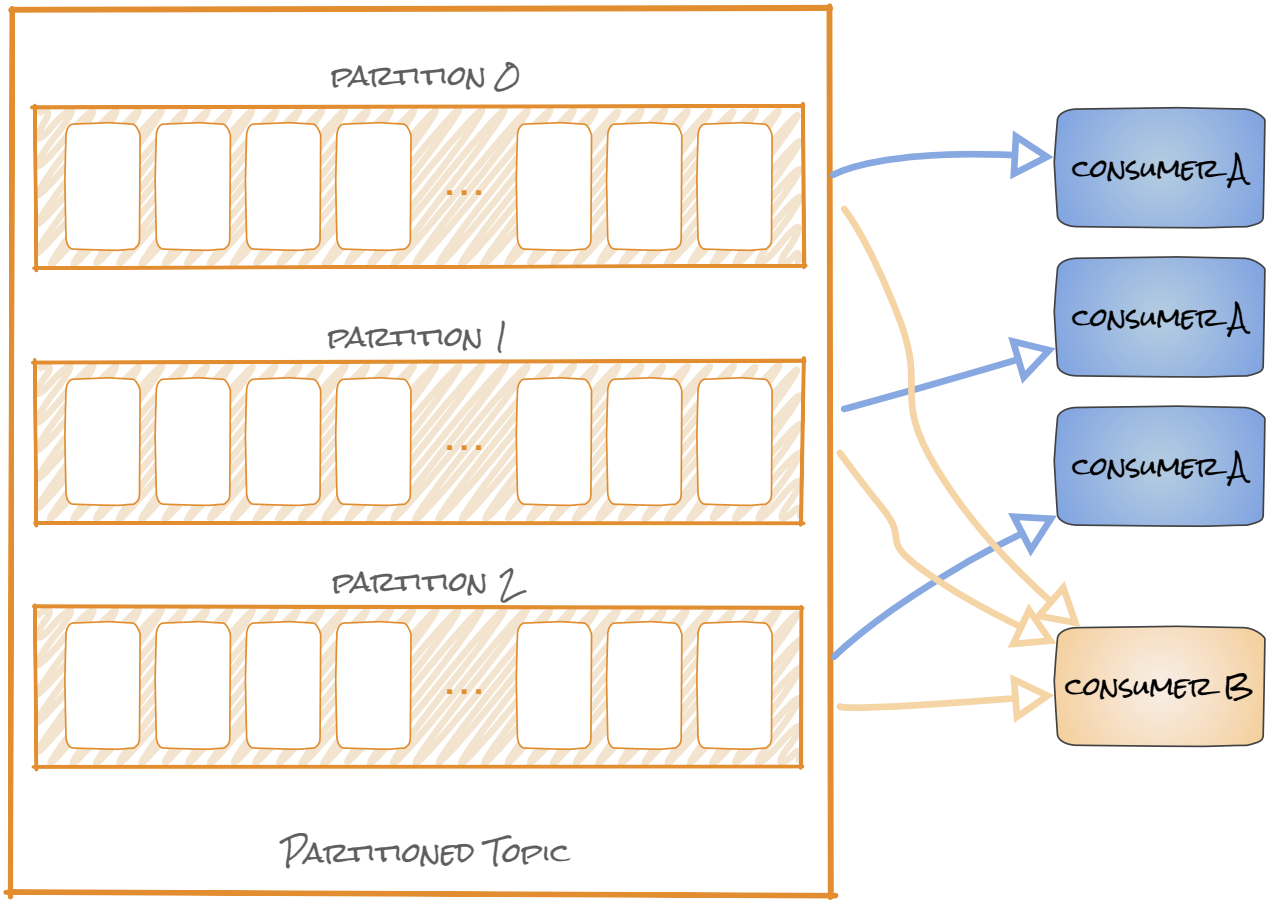

PRODUCER
PARTITION 0
PARTITION 1
PARTITION 2
PARTITIONED TOPIC

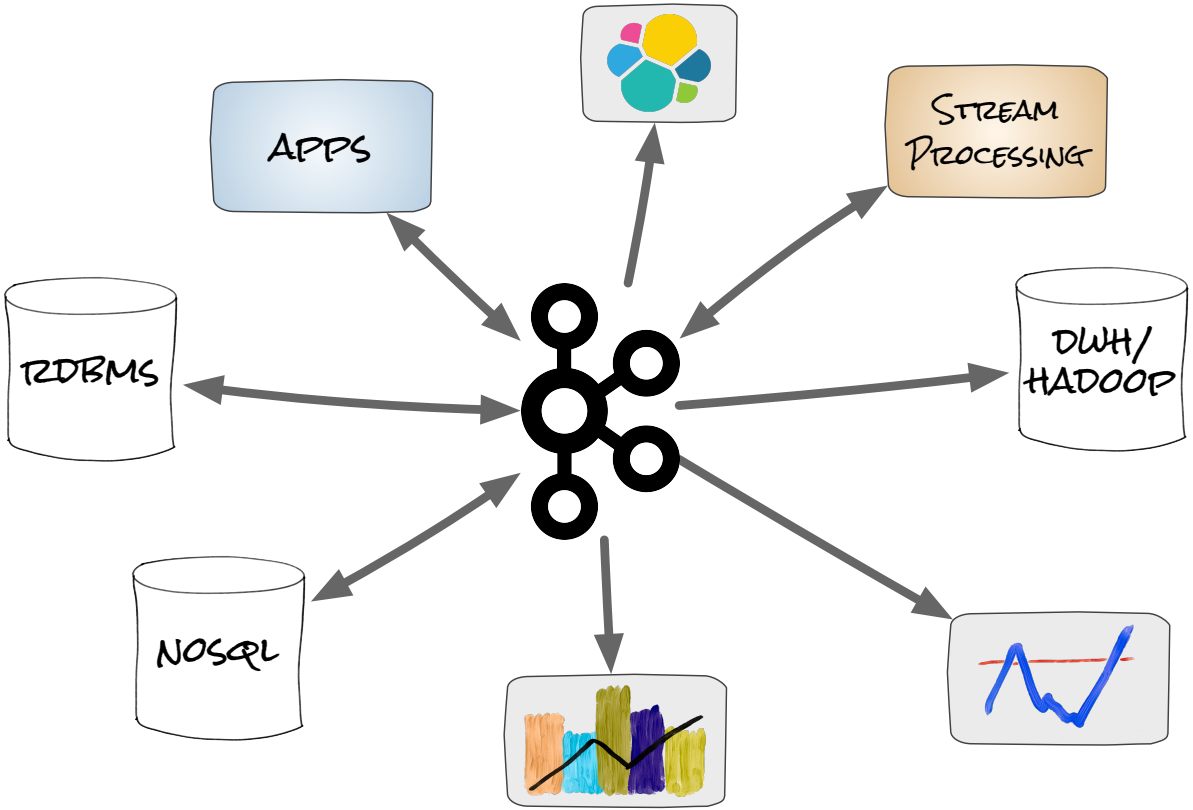

PARTITION 0
PARTITION 1
PARTITION 2
PARTITIONED TOPIC
CONSUMER A

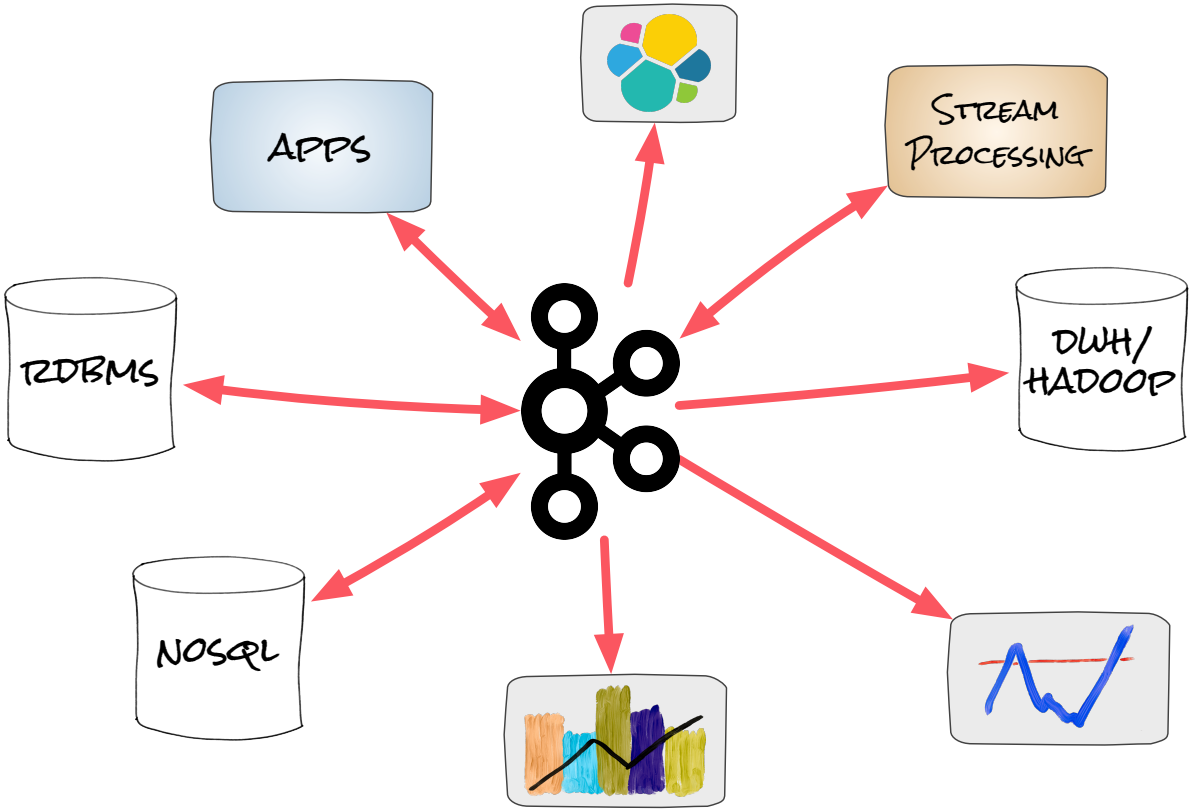

PARTITION 0
PARTITION 1
PARTITION 2
PARTITIONED TOPIC
CONSUMER A
CONSUMER B

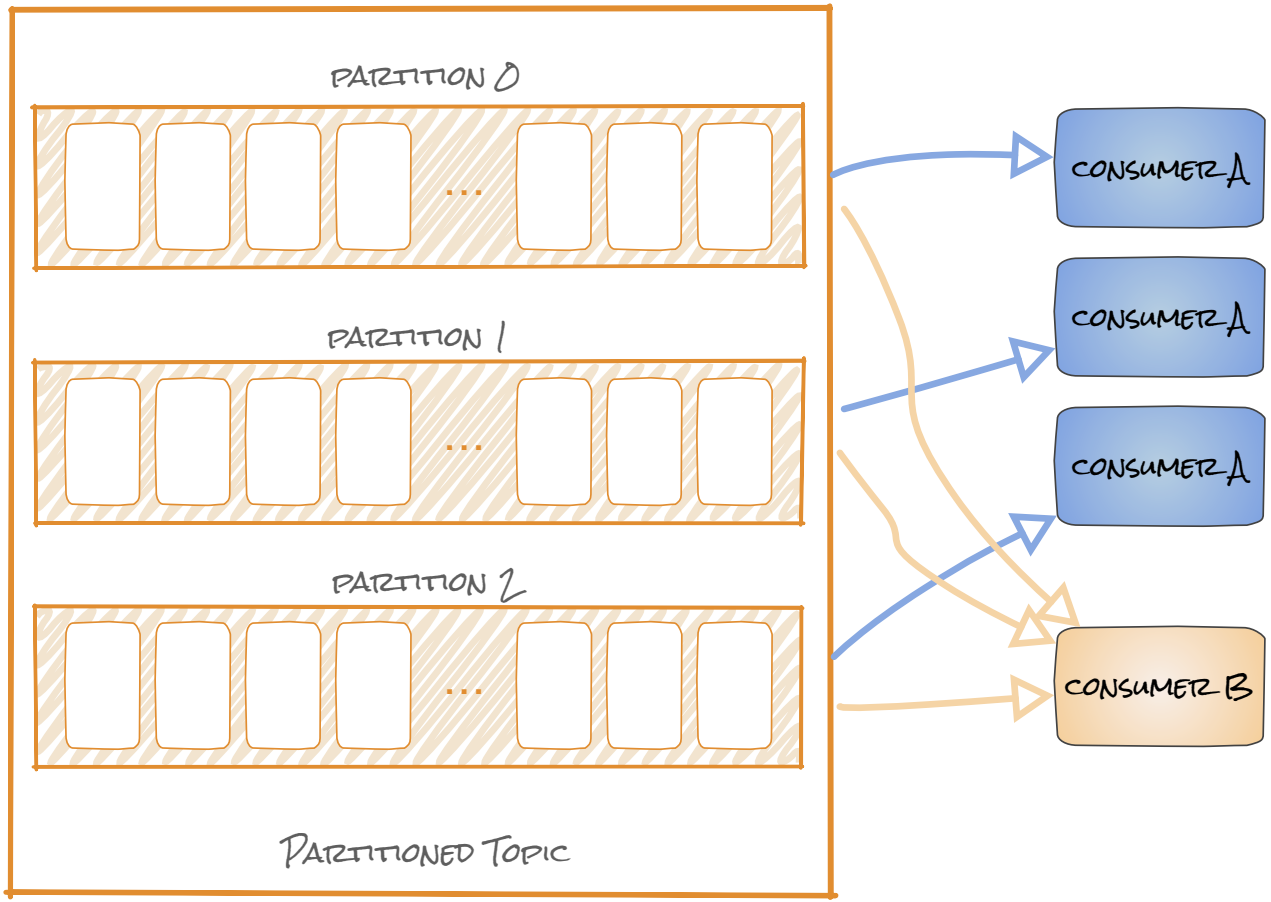

PARTITION 0
PARTITION 1
PARTITION 2
PARTITIONED TOPIC
CONSUMER A
CONSUMER B

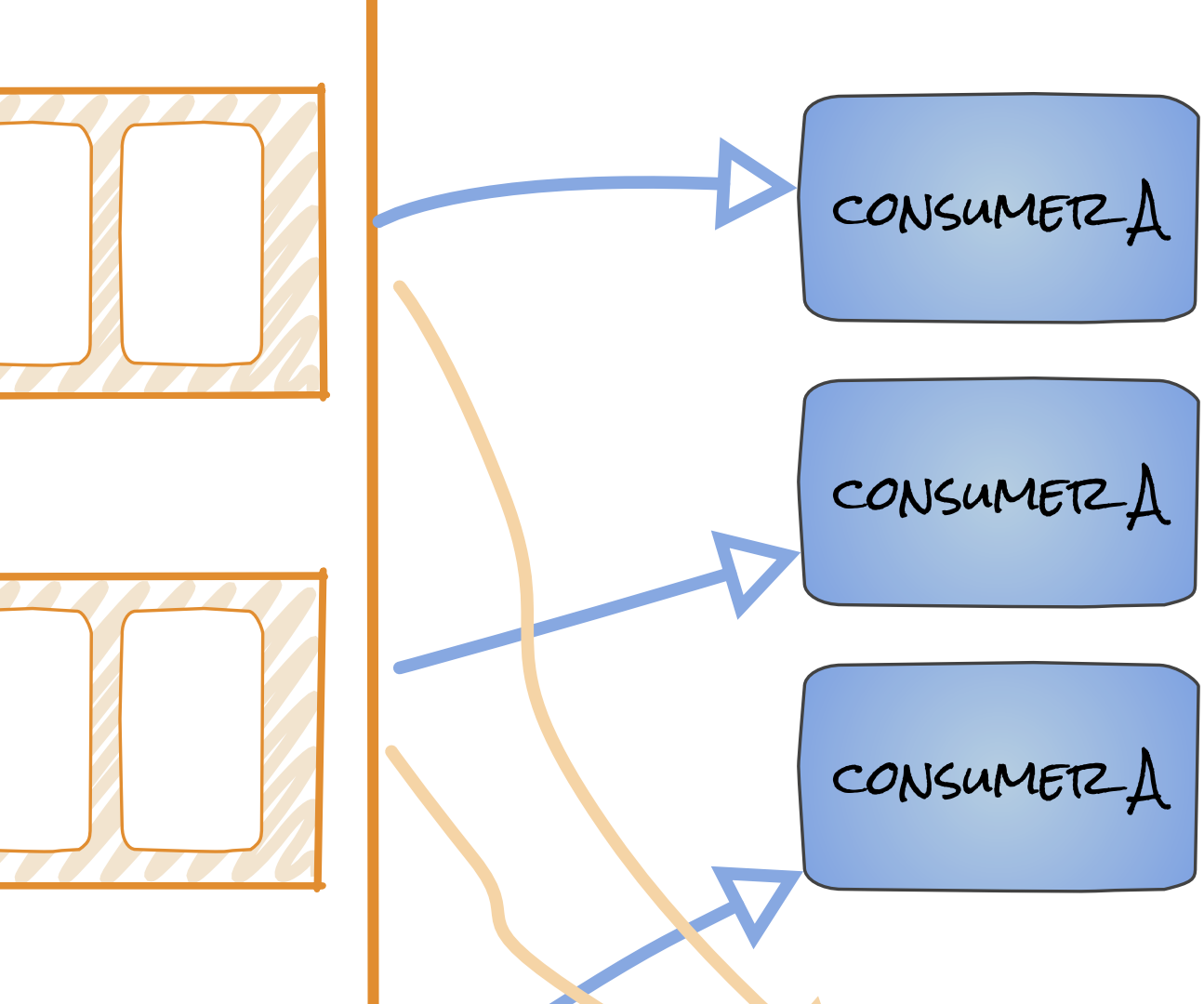

PARTITION 0
PARTITION 1
PARTITION 2
PARTITIONED TOPIC
CONSUMER A
CONSUMER A
CONSUMER B

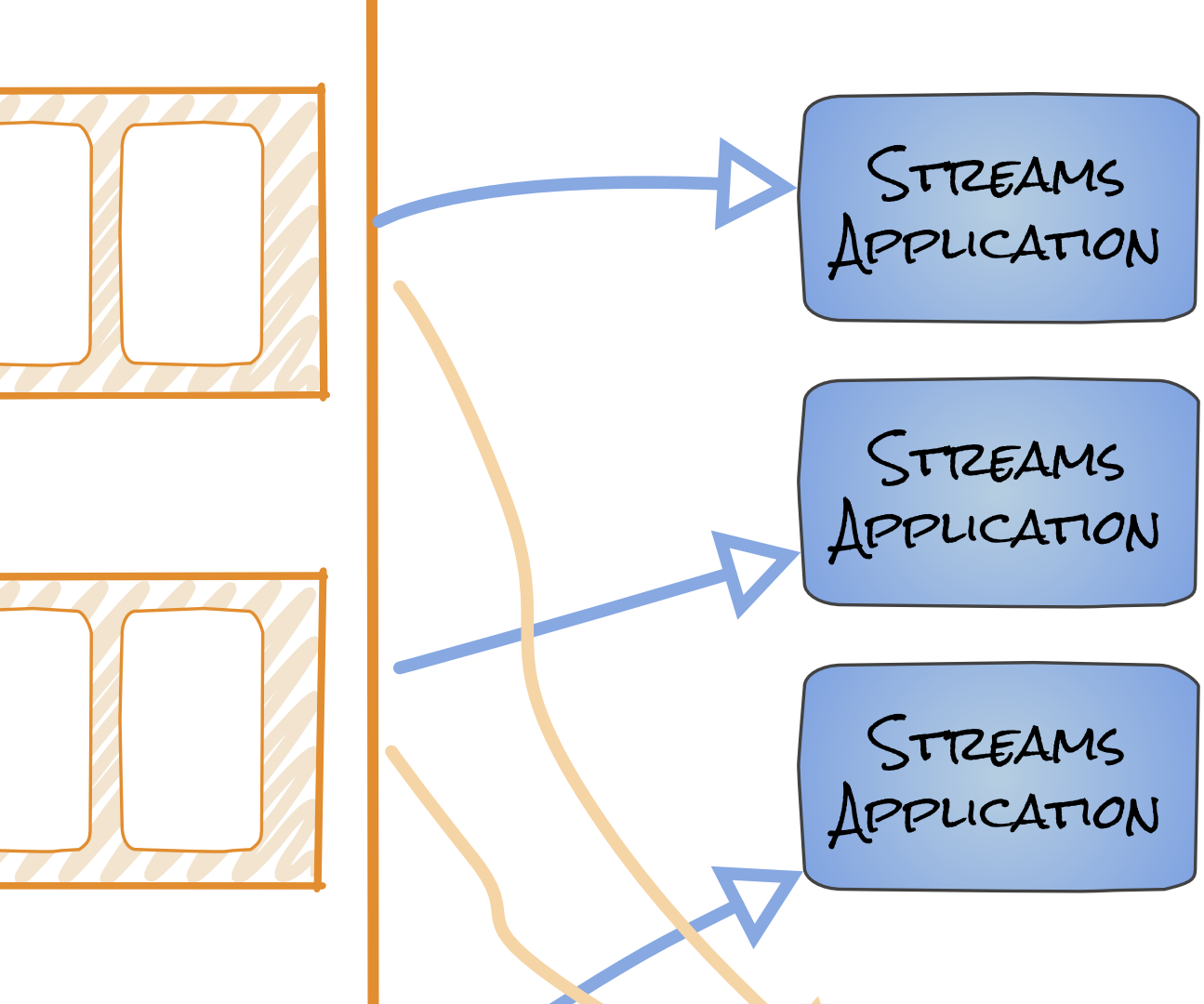

PARTITION 0
PARTITION 1
PARTITION 2
PARTITIONED TOPIC
CONSUMER A
CONSUMER A
CONSUMER A
CONSUMER B


APPS
STREAM PROCESSING
RDBMS
DWH/ HADOOP
NOSQL


APPS
STREAM PROCESSING
RDBMS
DWH/ HADOOP
NOSQL


PARTITION 0
PARTITION 1
PARTITION 2
PARTITIONED TOPIC
CONSUMER A
CONSUMER A
CONSUMER A
CONSUMER B


CONSUMER A
CONSUMER A
CONSUMER A

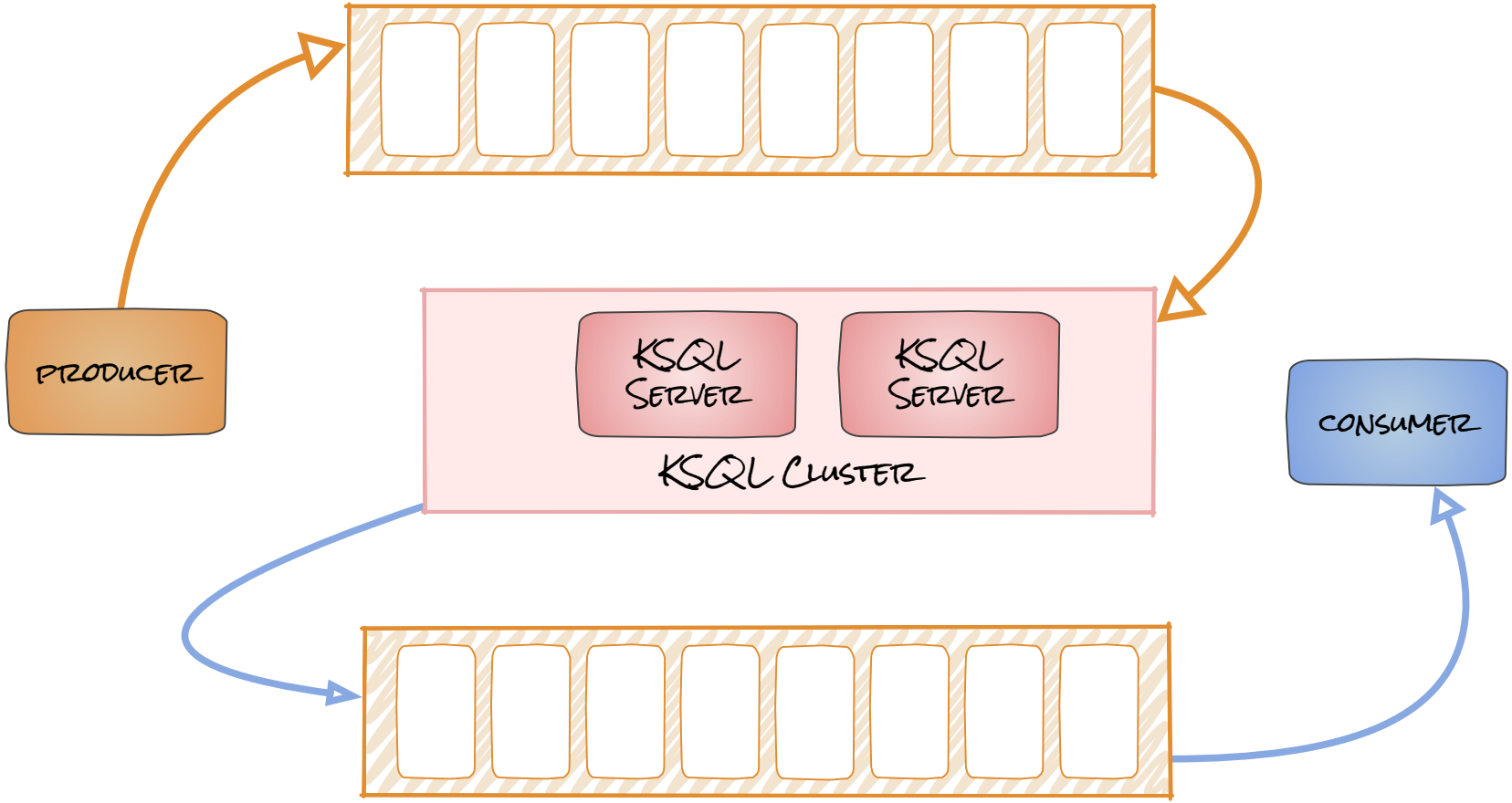

Streams Application
Streams Application
Streams Application

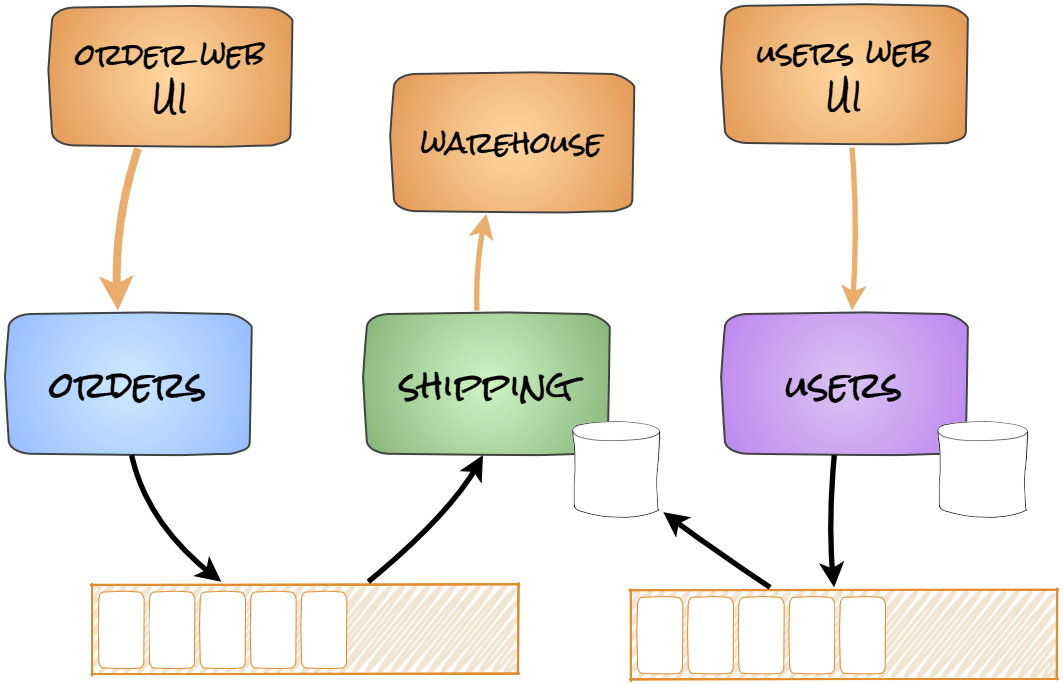
```
public static void main(String args[]) {

  Properties streamsConfiguration = getProperties(SCHEMA_REGISTRY_URL);

  final Map<String, String> serdeConfig =
      Collections.singletonMap(AbstractKafkaAvroSerDeConfig.SCHEMA_REGISTRY_URL_CONFIG,
                               SCHEMA_REGISTRY_URL);

  final SpecificAvroSerde<Movie> movieSerde = getMovieAvroSerde(serdeConfig);

  final SpecificAvroSerde<Rating> ratingSerde = getRatingAvroSerde(serdeConfig);

  final SpecificAvroSerde<RatedMovie> ratedMovieSerde = new SpecificAvroSerde<>();
  ratingSerde.configure(serdeConfig, false);

  StreamsBuilder builder = new StreamsBuilder();

  KTable<Long, Double> ratingAverage = getRatingAverageTable(builder);

  getRatedMoviesTable(builder, ratingAverage, movieSerde);

  Topology topology = builder.build();
  KafkaStreams streams = new KafkaStreams(topology, streamsConfiguration);
  Runtime.getRuntime().addShutdownHook(new Thread(streams::close));
  streams.start();
}

private static SpecificAvroSerde<Rating> getRatingAvroSerde(Map<String, String> serdeConfig) {
  final SpecificAvroSerde<Rating> ratingSerde = new SpecificAvroSerde<>();
  ratingSerde.configure(serdeConfig, false);
  return ratingSerde;
}

public static SpecificAvroSerde<Movie> getMovieAvroSerde(Map<String, String> serdeConfig) {
```

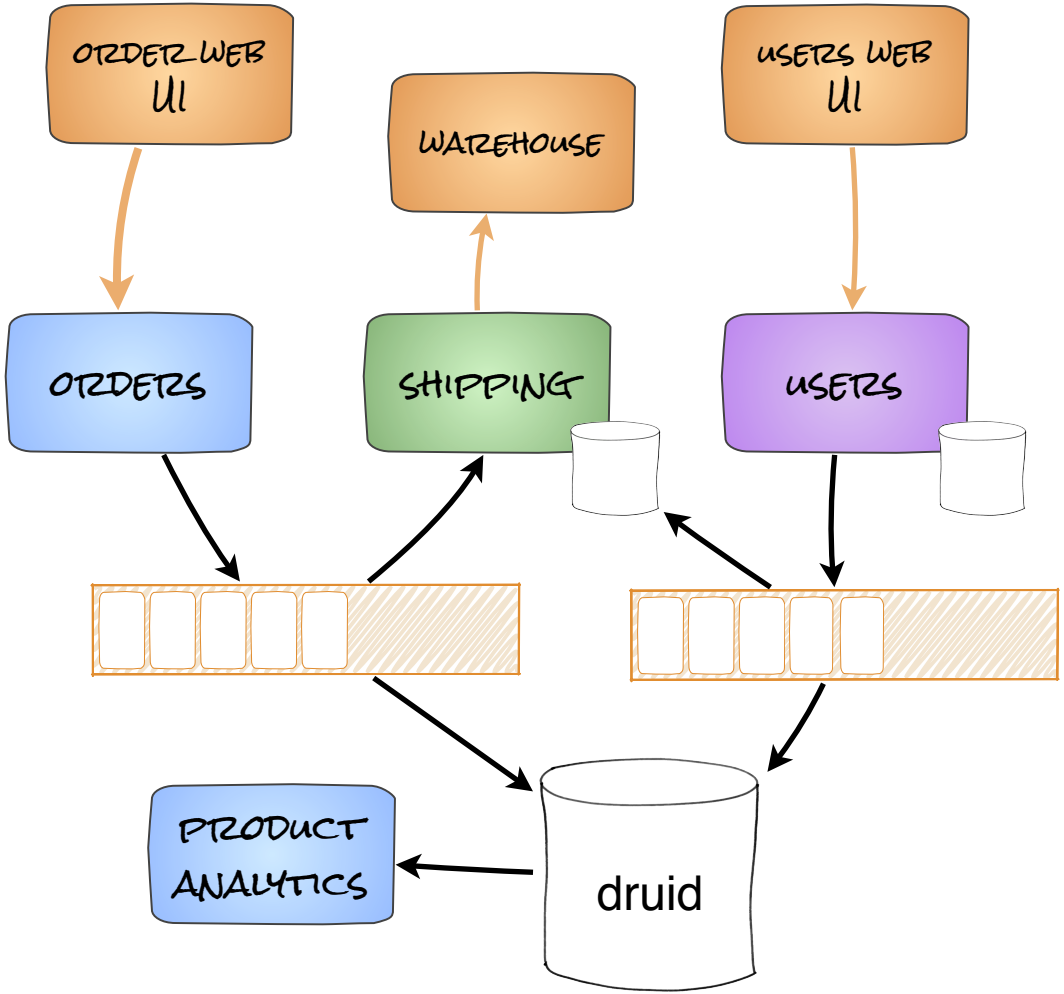
```
}

public static SpecificAvroSerde<Movie> getMovieAvroSerde(Map<String, String> serdeConfig) {
  final SpecificAvroSerde<Movie> movieSerde = new SpecificAvroSerde<>();
  movieSerde.configure(serdeConfig, false);
  return movieSerde;
}

public static KTable<Long, String> getRatedMoviesTable(StreamsBuilder builder,
                                                        KTable<Long, Double> ratingAverage,
                                                        SpecificAvroSerde<Movie> movieSerde) {

  builder.stream("raw-movies", Consumed.with(Serdes.Long(), Serdes.String()))
      .mapValues(Parser::parseMovie)
      .map((key, movie) -> new KeyValue<>(movie.getMovieId(), movie))
      .to("movies", Produced.with(Serdes.Long(), movieSerde));

  KTable<Long, Movie> movies = builder.table("movies",
                                             Materialized
                                                 .<Long, Movie, KeyValueStore<Bytes, byte[]>>as(
                                                     "movies-store")
                                                 .withValueSerde(movieSerde)
                                                 .withKeySerde(Serdes.Long())
  );

  KTable<Long, String> ratedMovies = ratingAverage
      .join(movies, (avg, movie) -> movie.getTitle() + "=" + avg);

  ratedMovies.toStream().to("rated-movies", Produced.with(Serdes.Long(), Serdes.String()));
  return ratedMovies;
}

public static KTable<Long, Double> getRatingAverageTable(StreamsBuilder builder) {
  KStream<Long, String> rawRatings = builder.stream("raw-ratings",
                                                    Consumed.with(Serdes.Long(),
                                                                  Serdes.String()));
  KStream<Long, Rating> ratings = rawRatings.mapValues(Parser::parseRating)
```

```
  return ratedMovies;
}

public static KTable<Long, Double> getRatingAverageTable(StreamsBuilder builder) {
  KStream<Long, String> rawRatings = builder.stream("raw-ratings",
                                                    Consumed.with(Serdes.Long(),
                                                                  Serdes.String()));
  KStream<Long, Rating> ratings = rawRatings.mapValues(Parser::parseRating)
      .map((key, rating) -> new KeyValue<>(rating.getMovieId(), rating));

  KStream<Long, Double> numericRatings = ratings.mapValues(Rating::getRating);

  KGroupedStream<Long, Double> ratingsById = numericRatings.groupByKey();

  KTable<Long, Long> ratingCounts = ratingsById.count();
  KTable<Long, Double> ratingSums = ratingsById.reduce((v1, v2) -> v1 + v2);

  KTable<Long, Double> ratingAverage = ratingSums.join(ratingCounts,
                                                       (sum, count) -> sum / count.doubleValue(),
                                                       Materialized.as("average-ratings"));
  ratingAverage.toStream()
      /*.peek((key, value) -> { // debug only
        System.out.println("key = " + key + ", value = " + value);
      })*/
      .to("average-ratings");
  return ratingAverage;
}
```

```
CREATE TABLE movie_ratings AS
            SELECT title,
                   SUM(rating)/COUNT(rating) AS avg_rating,
                   COUNT(rating) AS num_ratings
              FROM ratings
                   LEFT OUTER JOIN movies
                   ON ratings.movie_id = movies.movie_id
            GROUP BY title;
```

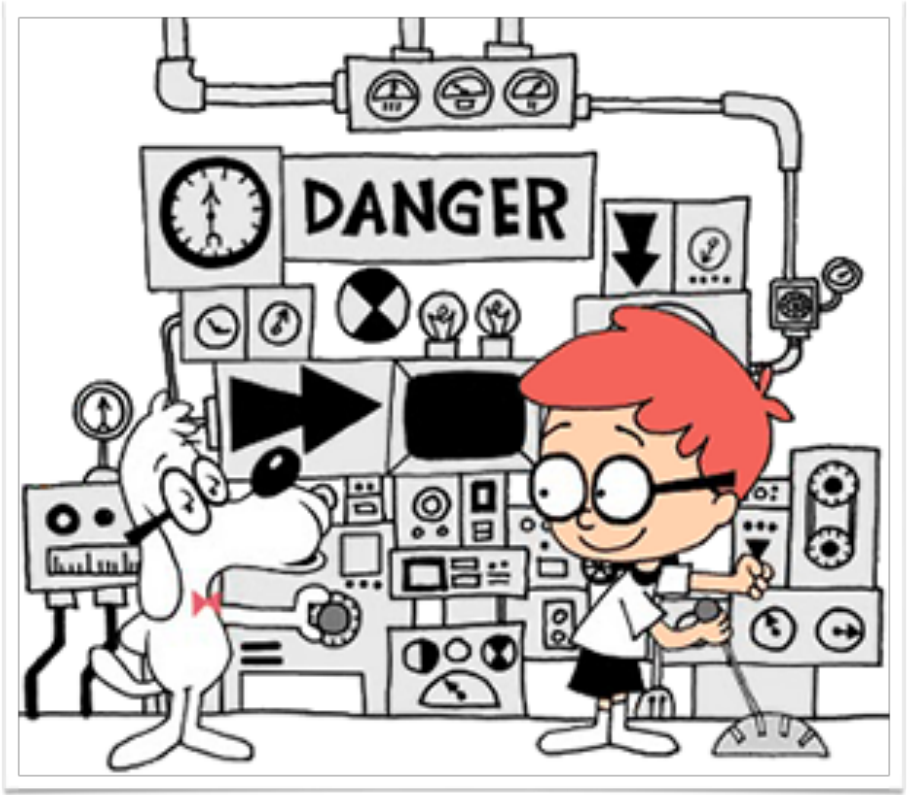

PRODUCER
KSQL SERVER
KSQL SERVER
KSQL CLUSTER
CONSUMER


ORDER WEB UI
WAREHOUSE
USERS WEB UI
ORDERS
SHIPPING
USERS


ORDER WEB UI
WAREHOUSE
USERS WEB UI
ORDERS
SHIPPING
USERS
PRODUCT ANALYTICS
druid

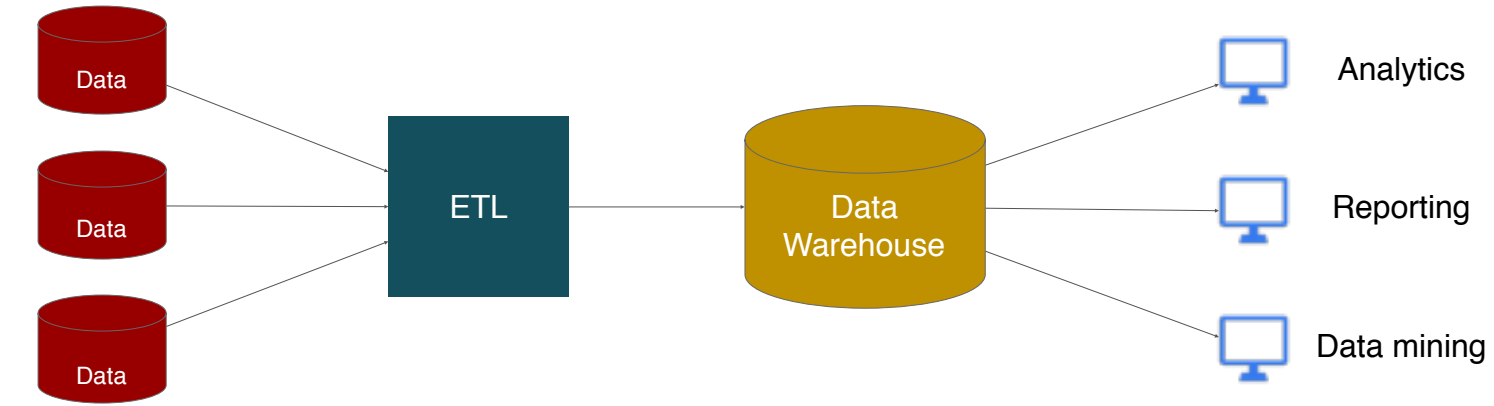
What the heck is Apache Druid and Why Should I Care?

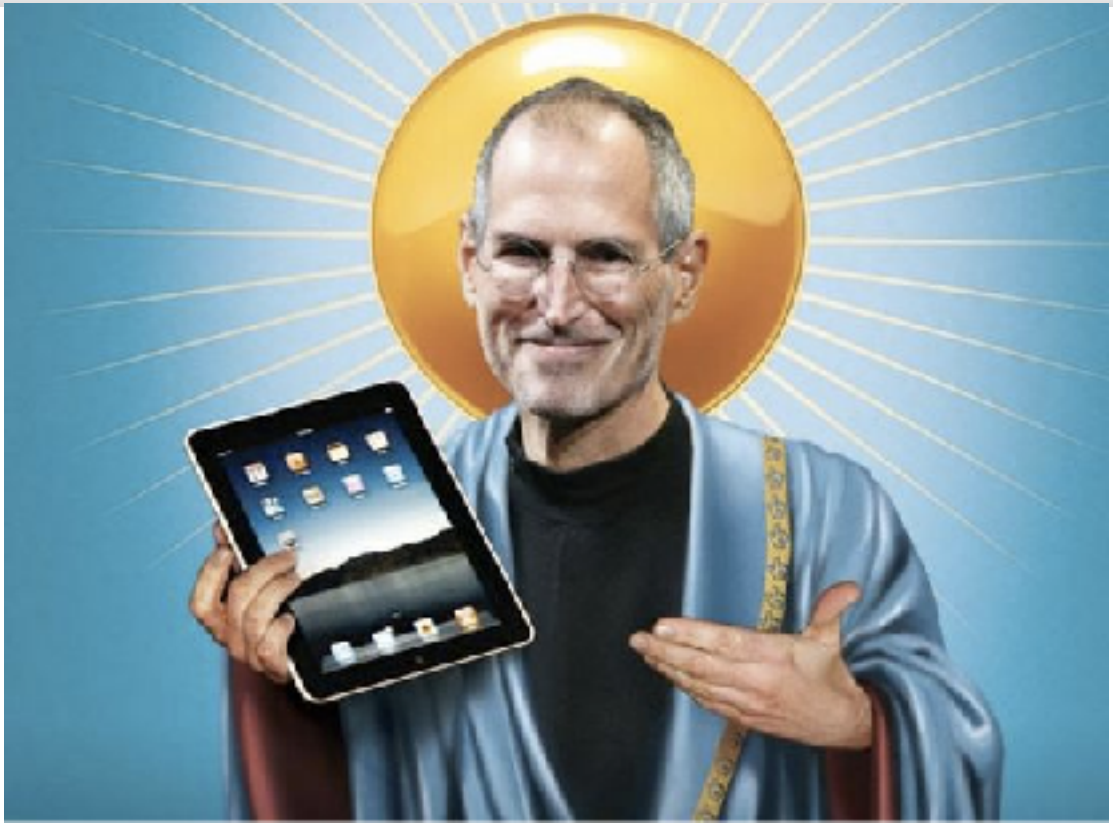
Rachel

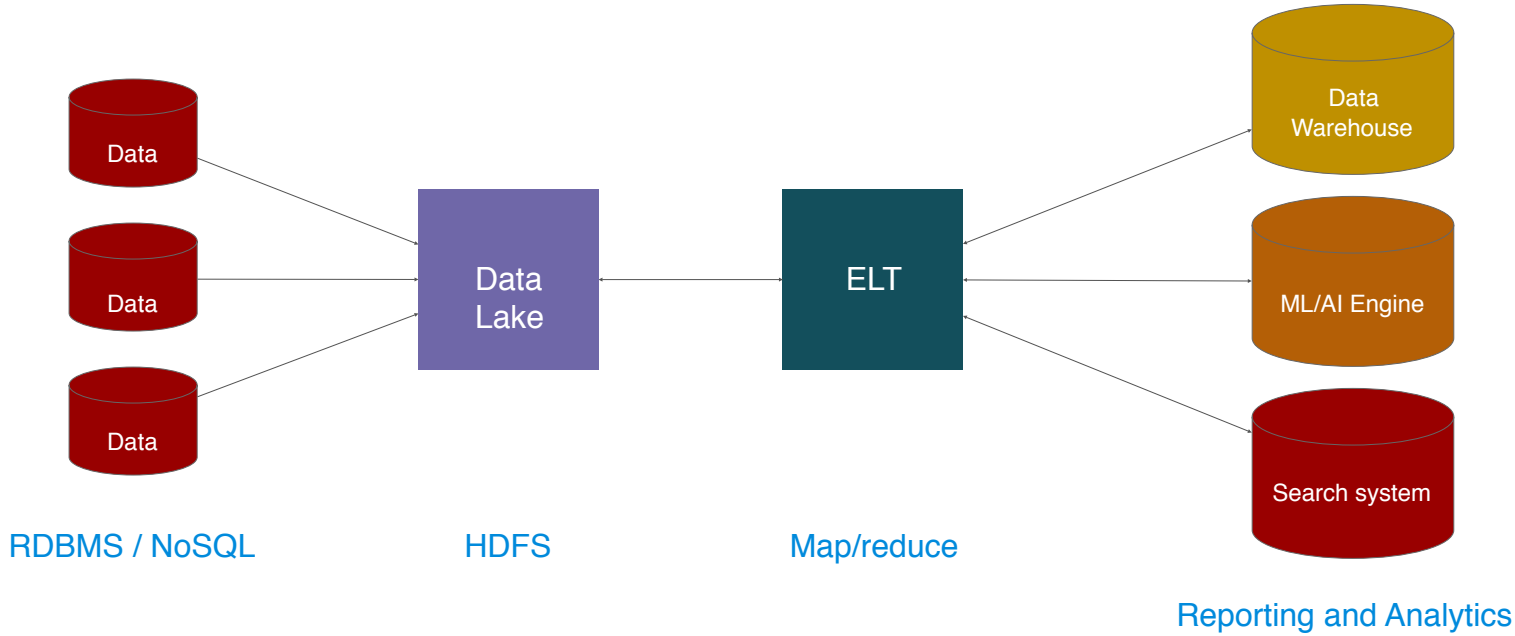


Tim

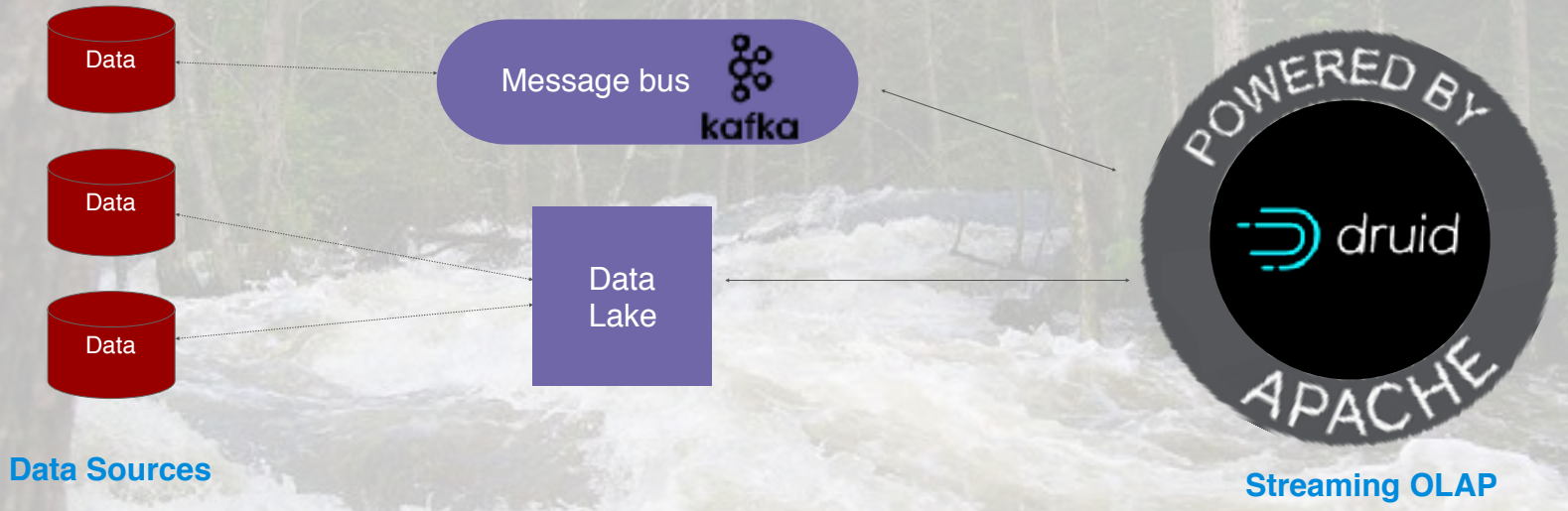

You're
Welcome
Al Gore

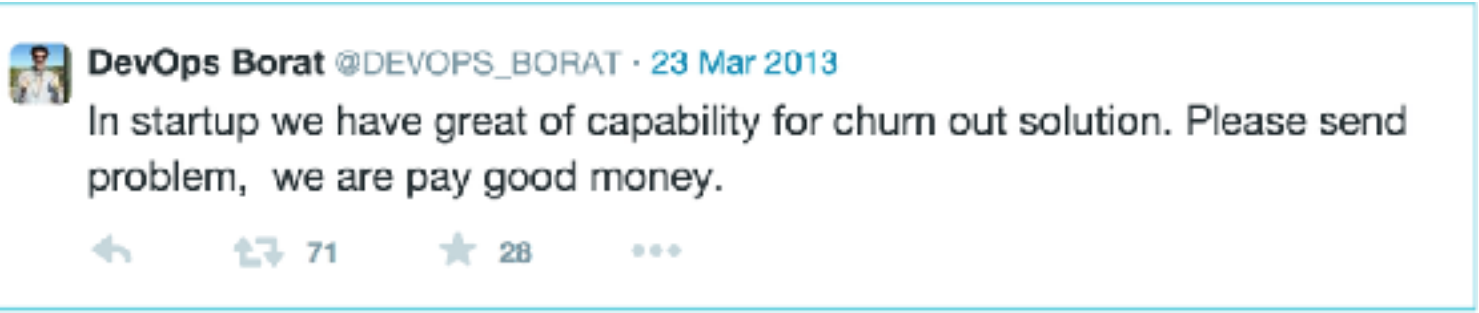

Data
Data
Data
ETL
Data Warehouse
Analytics
Reporting
Data mining
Data Sources
Some Code
Usually an RDBMS
Querying

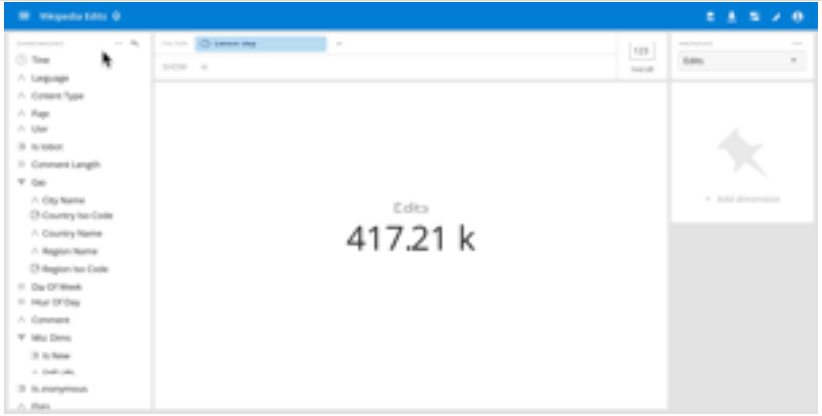



Data
Data
Data
Data Lake
ELT
Data Warehouse
ML/AI Engine
Search system
RDBMS / NoSQL
HDFS
Map/reduce
Reporting and Analytics


NOW


Data
Data
Data
Message bus
kafka
Data Lake
POWERED BY
druid
APACHE
Data Sources
Streaming OLAP


DevOps Borat @DEVOPS_BORAT · 23 Mar 2013
In startup we have great of capability for churn out solution. Please send problem, we are pay good money.
71
28


Edits
417.21 k

Typical Big Data++ Challenges

- Scale: when data is large, we need a lot of servers
- Speed: aiming for sub-second response time
- Complexity: too much fine grain to precompute
- High dimensionality: 10s or 100s of dimensions
- Concurrency: many users and tenants
- Freshness: load from streams

Search platform

- Real-time ingestion
- Flexible schema
- Full text search

Timeseries database

- Optimized storage for time-based datasets
- Time-based functions

OLAP

- Batch ingestion
- Efficient storage
- Fast analytic queries

Search platform

- Real-time ingestion
- Flexible schema
- Full text search

Timeseries database

- Optimized storage for time-based datasets
- Time-based functions

OLAP

- Batch ingestion
- Efficient storage
- Fast analytic queries

high performance analytics **database** for event-driven data

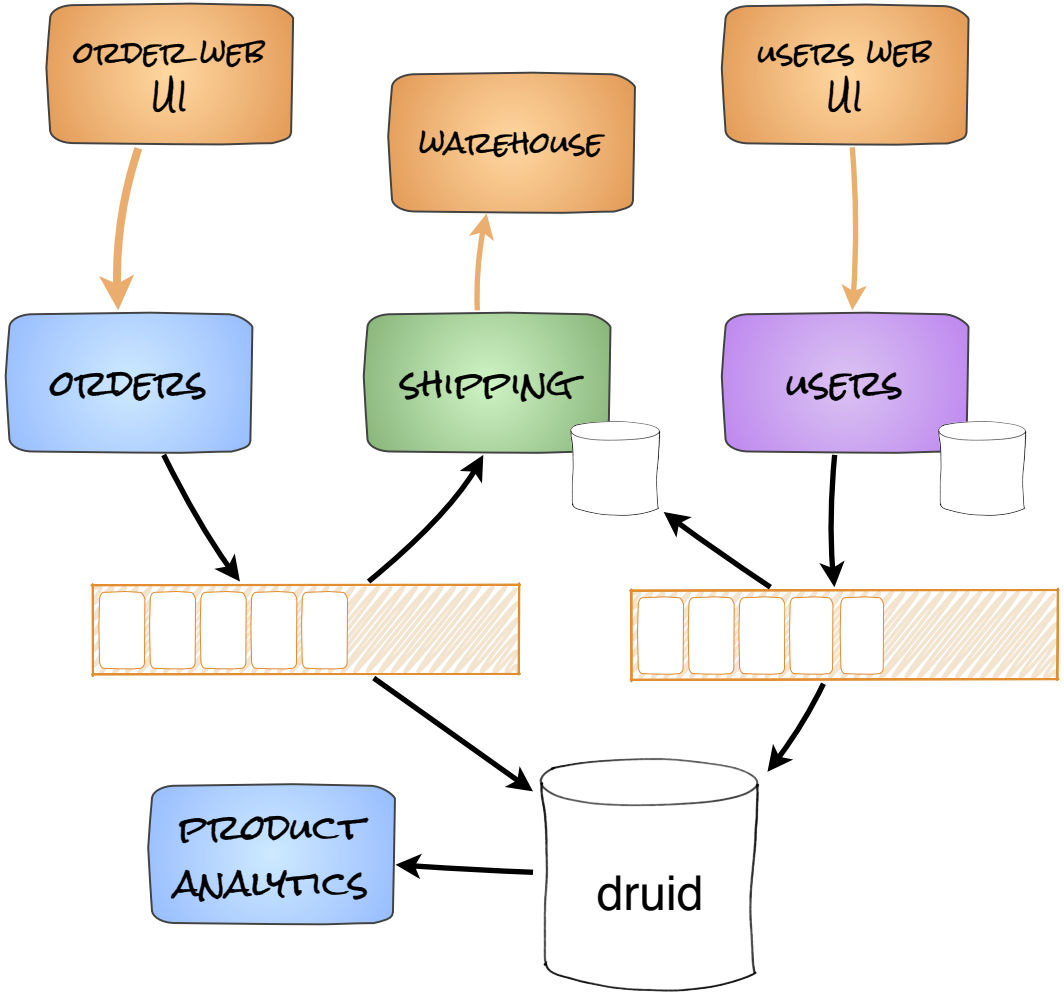
Druid Use Cases in the Wild

1. Digital Advertising - Publishers, Advertisers, Exchanges
2. User Event Analytics- Clickstream, QoS, Usage
3. Network Telemetry
4. Lots and Lots of Data- IoT, Product Analytics, Fraud

Gratuitous Customer Quote

"The performance is great ... some of the tables that we have internally in Druid have **billions and billions of events** in them, and we're scanning them in **under a second**."

Source: https://www.infoworld.com/article/2949168/hadoop/yahoo-struts-its-hadoop-stuff.html


ORDER WEB UI
WAREHOUSE
USERS WEB UI
ORDERS
SHIPPING
USERS
PRODUCT ANALYTICS
druid


NOW DEMO
VERY NICE
imgflip.com

You can Druid too!

Druid community site: https://druid.apache.org/

Imply distribution: https://imply.io/get-started

Contribute: https://github.com/apache/druid


confluent