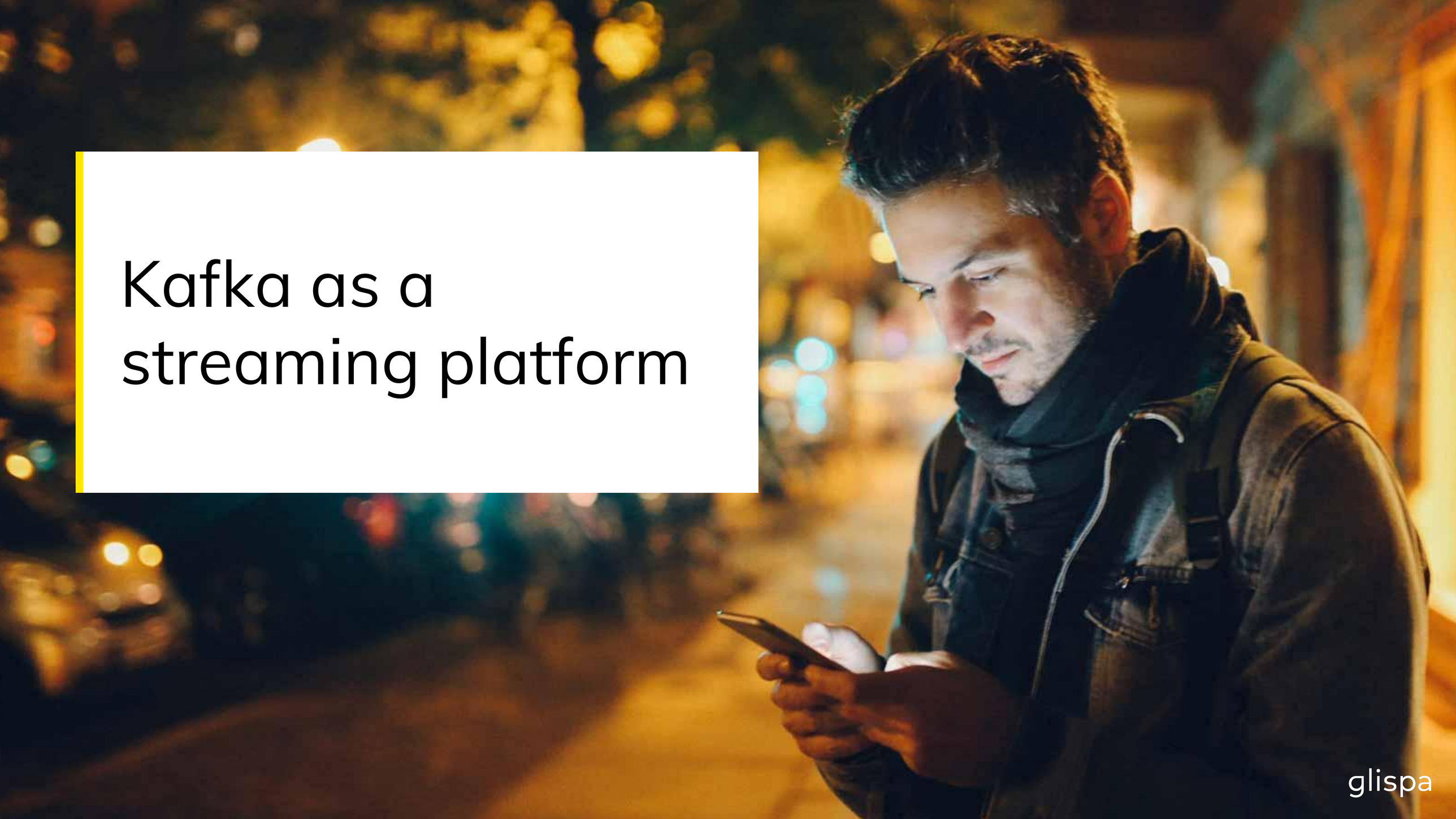
Kafka as a streaming platform



glispa

Glispa in a nutshell

Mobile ad tech made simple

Our mission

Empowering you with connected solutions

Our people

50 nationalities speaking **45** different languages

Our tech

1000+ instances
3 PB+/month outbound traffic
Deployed in **5** glispa datacenters + AWS

glispa

Why streaming your data ?

Batch vs. Stream processing

You need to spread the data across multiple components

Have your data available and in-sync everywhere

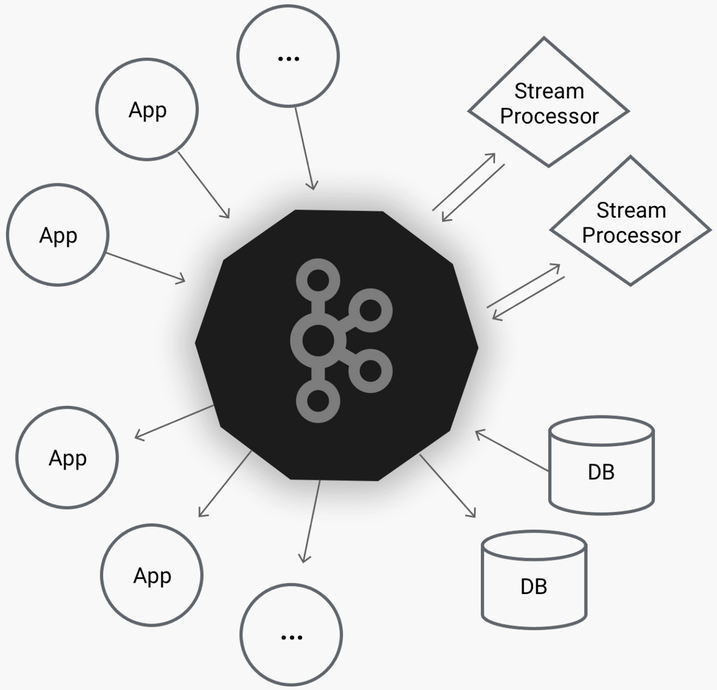
You need results as soon as possible

Some business use cases require fast results

You need performance

When hitting disk is problematic

Kafka ecosystem



Kafka cluster

The backbone and data storage

Kafka Connect

Move data in, move data out

Kafka Streams

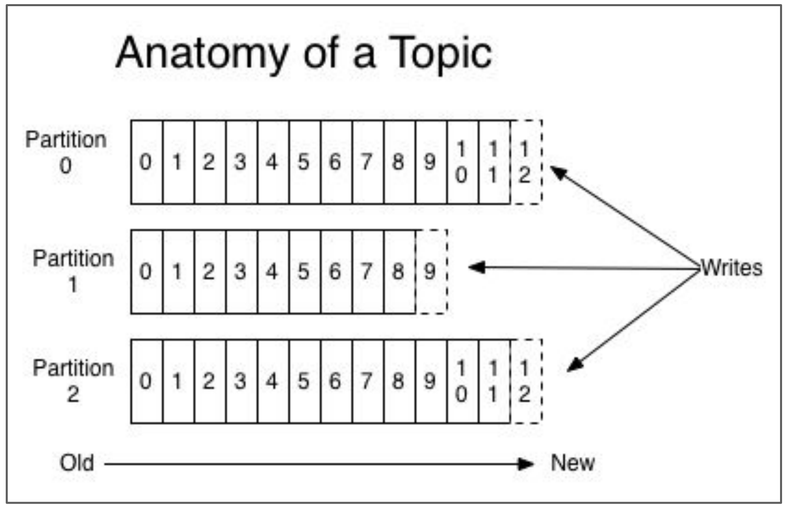
Process data

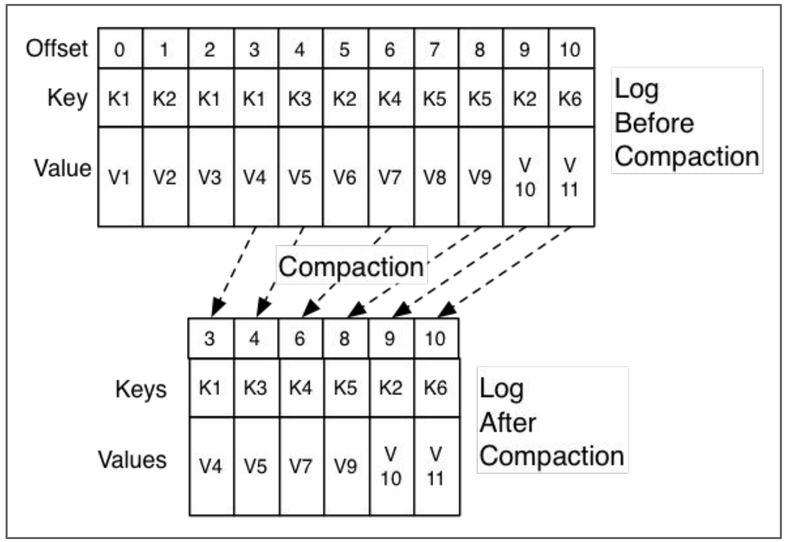
Kafka

The backbone

Data as a log

- Topics, partitions, segments
- Clean up policies
  - deletion for fact tables
  - compaction for dimension tables





glispa

Kafka

The backbone

The broker

- Manages log files, in cluster
- Offers a TCP API

-> what about performance ?
-> what about sizing ?

The producer

- Choose the partition to write to
- Avoid duplicates* (since 0.11.0.0)

The consumer

- Manages consumed offsets
- Works in group

Kafka Connect

Move data in, Move data out

The workers

- Plain java application
- Use kafka topics for internal task scheduling
- Offer HTTP API for managing task

-> How to install and run it ?

Source connectors

- Use a specific kafka topic to store state
- Based on subscription, oplog or polling

Sink connectors

- Use regular kafka consumer group

Transformation and schema

- Schema management (for example with Avro)
- Small transformation system

Kafka Connect

JDBC Source example

```
"connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
"key.converter": "org.apache.kafka.connect.storage.StringConverter",
"value.converter": "org.apache.kafka.connect.json.JsonConverter",
"key.converter.schemas.enable": false,
"value.converter.schemas.enable": false,
"tasks.max": 1,
"connection.url": "${CONNECT_JDBC_URL}",
"topic.prefix": "users",
"mode": "timestamp+incrementing",
"incrementing.column.name": "uid",
"timestamp.column.name": "update_dt",
"query": "SELECT uid, update_dt, email, name FROM members",
"transforms": "ValueToKey,ReplaceField,ExtractField",
"transforms.ValueToKey.type":
"org.apache.kafka.connect.transforms.ValueToKey",
"transforms.ValueToKey.fields": "uid",
"transforms.ReplaceField.type":
"org.apache.kafka.connect.transforms.ReplaceField$Value",
"transforms.ReplaceField.blacklist": "uid,update_dt"
```

Kafka Connect

Elastic sink example

```
"connector.class":
"io.confluent.connect.elasticsearch.ElasticsearchSinkConnector",
"key.converter": "org.apache.kafka.connect.storage.StringConverter",
"value.converter": "org.apache.kafka.connect.json.JsonConverter",
"key.converter.schemas.enable": false,
"value.converter.schemas.enable": false,
"tasks.max": 2,
"topics": "users,products,campaigns,publishers,advertisers,
exchange-rates,post-conversion-mappings,theygets,wegets,creatives",
"connection.url": "http://bom-elk:9200",
"schema.ignore": true,
"behavior.on.null.values": "delete",
"behavior.on.malformed.documents": "warn",
"drop.invalid.message": "true",
"type.name": "_doc"
```

Kafka Streams

The workers

- Plain java application
- Only need Kafka
- One thread per partition

-> How to install and run it ?

Defining topologies

- Processor language (like Storm) with Processor API
- High level language (like Spark RDD) with Streams DSL
- SQL language (like Spark SQL) with KSQL

Process data

```
StreamsBuilder builder = new StreamsBuilder();
KStream<String, String> textLines = builder.stream("TextLinesTopic");
KTable<String, Long> wordCounts = textLines
    .flatMapValues(textLine -> Arrays.asList(textLine.toLowerCase().split("\\W+")))
    .groupBy((key, word) -> word)
    .count(Materialized.<String, Long, KeyValueStore<Bytes, byte[]>>as("counts-store"));
wordCounts.toStream().to("WordsWithCountsTopic", Produced.with(Serdes.String(), Serdes.Long()));
```

Kafka Streams

Process data

Stateless transformations

- map, filter, flatMap, branch
- Based on kafka consumers

Shuffling

- groupBy
- Based on a new kafka topic
- Producers will hash by key
- Consumers will consume the new topic

Stateful transformation

- aggregate, count, reduce
- State is stored as a kafka topic
- Locally materialized in memory or in rocksdb

Tips

Do it right !

Kafka

- Carefully choose how to serialize your data
- Carefully tune your topics (compression, cleanup policies)

Kafka Connect

- Do not put logic in the connectors
- Open source connectors !
- You might need to wait for newer versions

Kafka Streams

- Carefully choose how to serialize your state
- Automate your app management
- Manually setup the intermediate topics
- Store enough “facts” to be able to rebuild a valuable state

Questions ?

Vincent Maurin
Director of Engineering

vincent.maurin@glispa.com