Cloud Chats
by aiven

Apache Kafka 101: The Why, What, and How

John Hammink
Developer Advocate

Chris Gwilliams
Solutions Architect

aiven

Aiven brings the best Open Source data technologies to all public clouds


Kafka


PostgreSQL


Elasticsearch


Cassandra


Grafana


Redis


InfluxDB


MySQL

Coming soon...

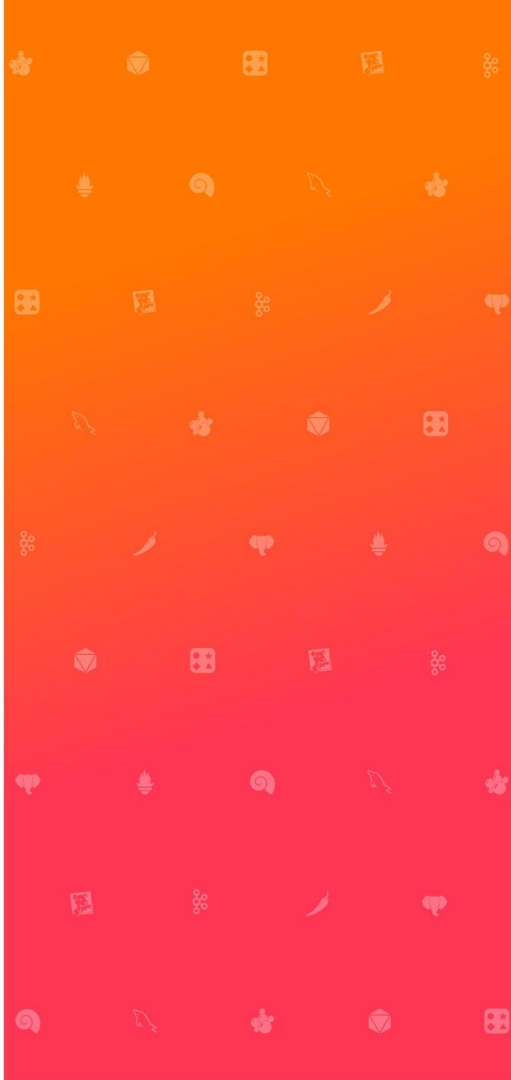

Clickhouse


M3


Flink

Agenda

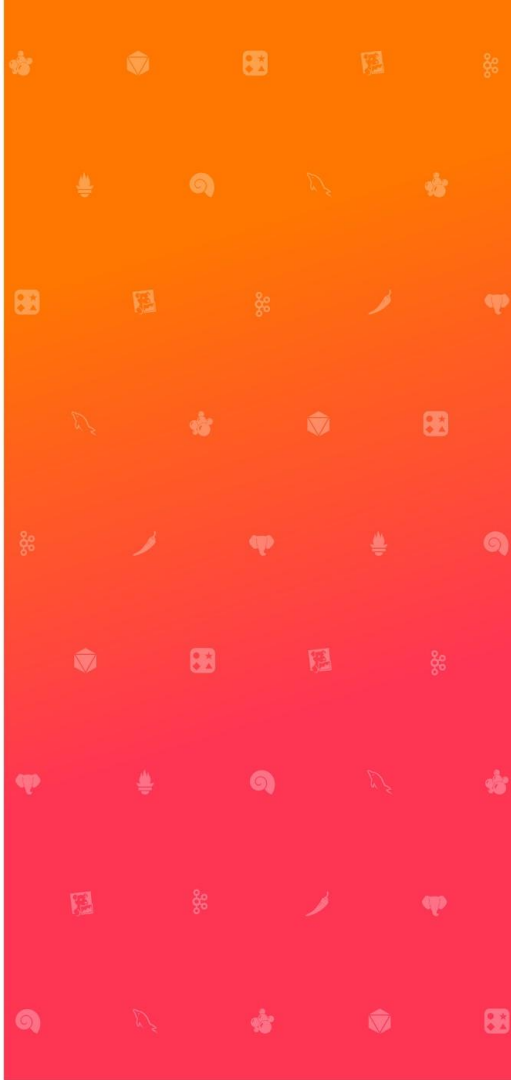
- Introduction
- Kafka -
  - WHY?
  - WHAT?
  - HOW?
- Kafka ecosystem
- DEMO
- Q & A (aka HUH?)



Background: the WHY of Kafka

History of Apache Kafka

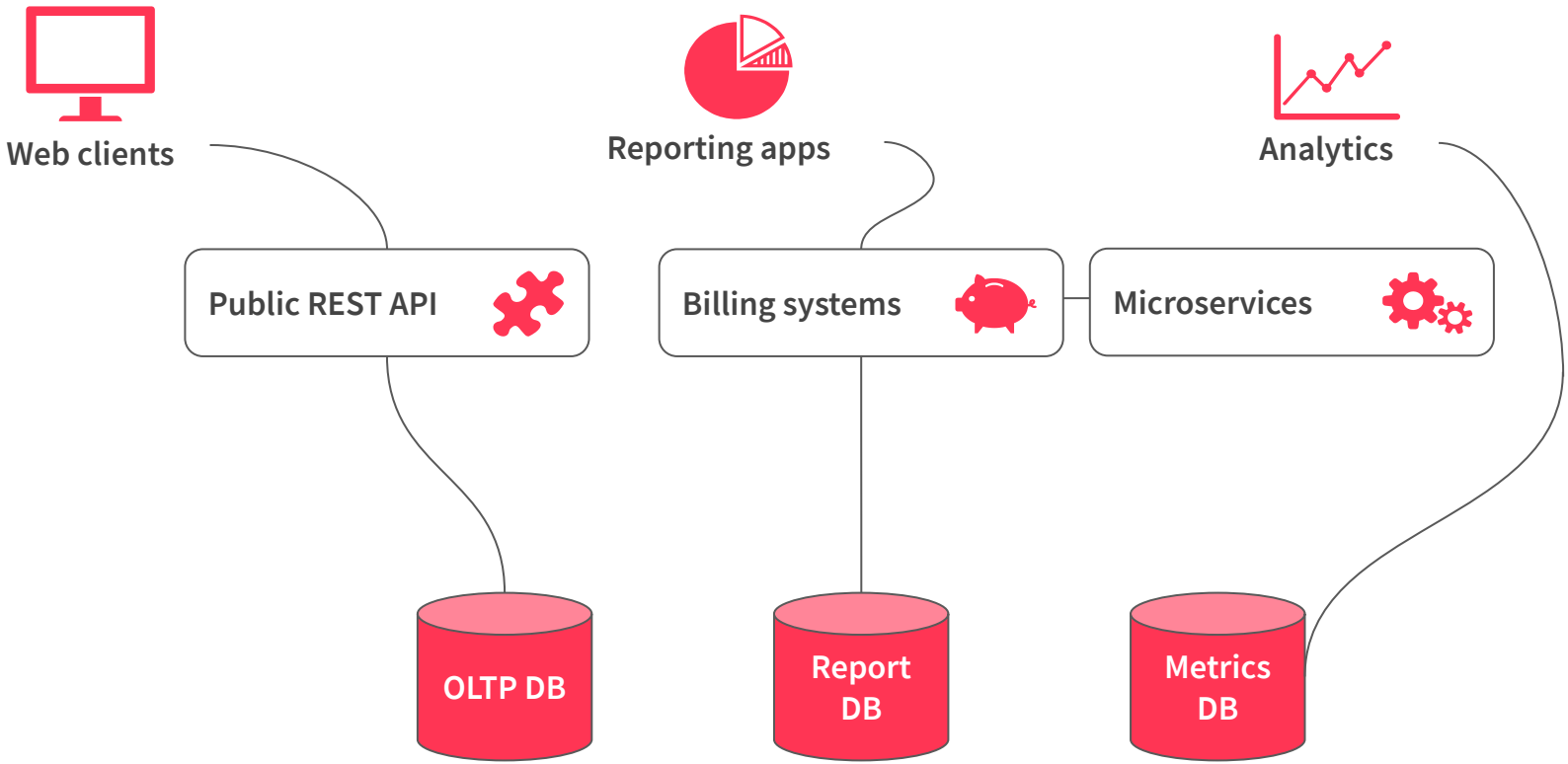
- shift away from “monolithic” programs to those that must pass data between ***senders*** and ***receivers***
- eventually, systems became **distributed**
- imagine the complexity with **millions of nodes** sending items back and forth like **status updates**, **presence**, **login states**, among other things. Imagine the volume of messages here.
- complexity = **n * n**
- **Also, decoupling...**



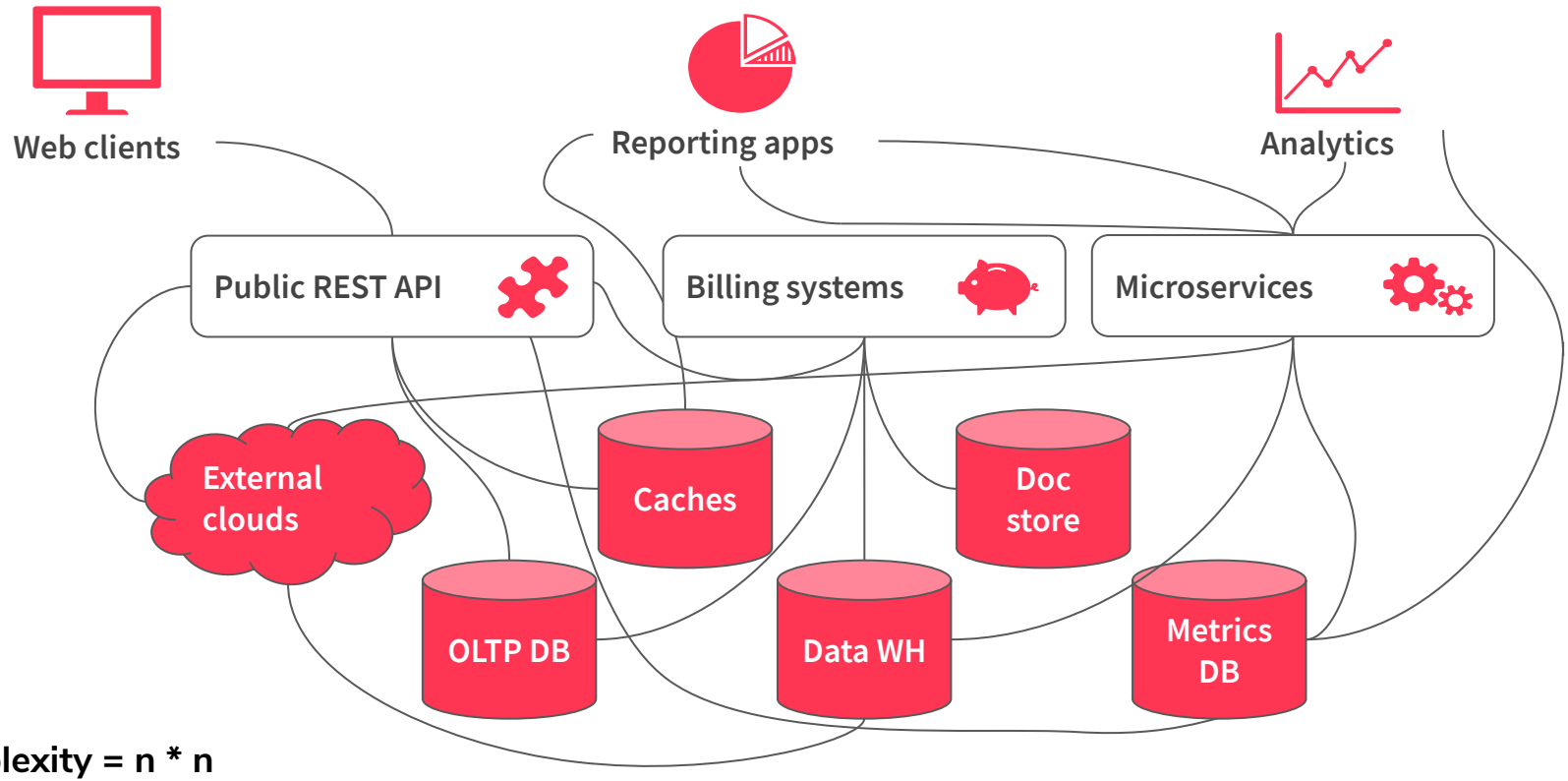
History, part II

Alas, a middle step:

“Traditional” data flow model



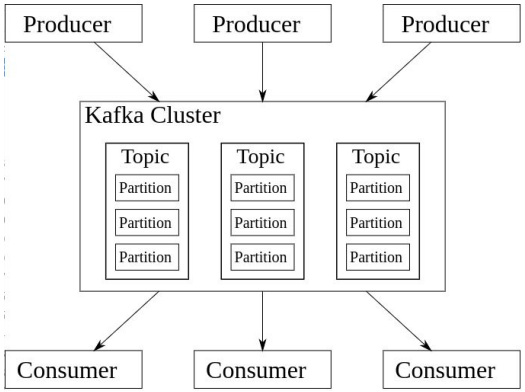
“Traditional” data flow model



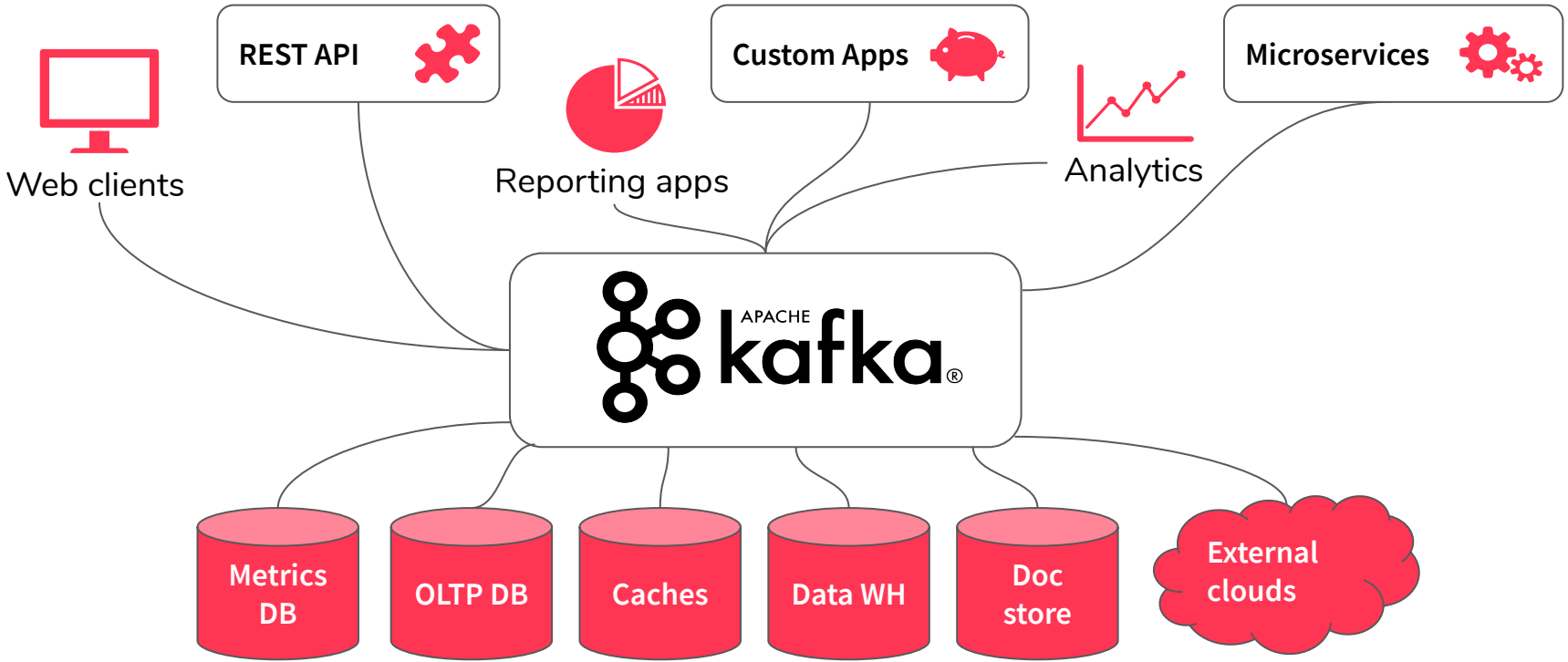
complexity = n * n

Enter: Apache Kafka!

- Developed at LinkedIn starting in 2010
- open-sourced to Apache in early 2011
- improved on the pub/sub architecture
- complexity = **n + n**



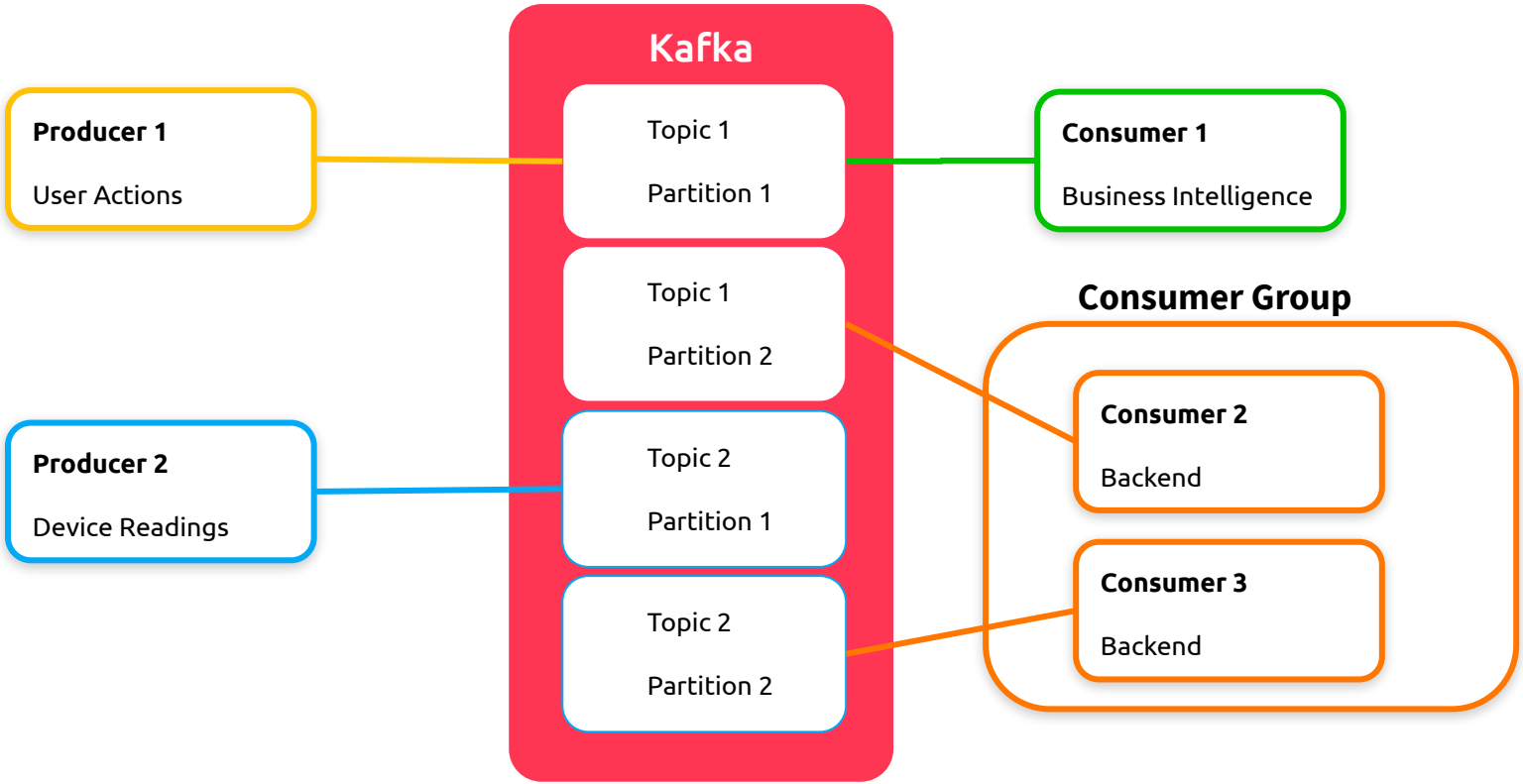
Kafka-centric data flow model



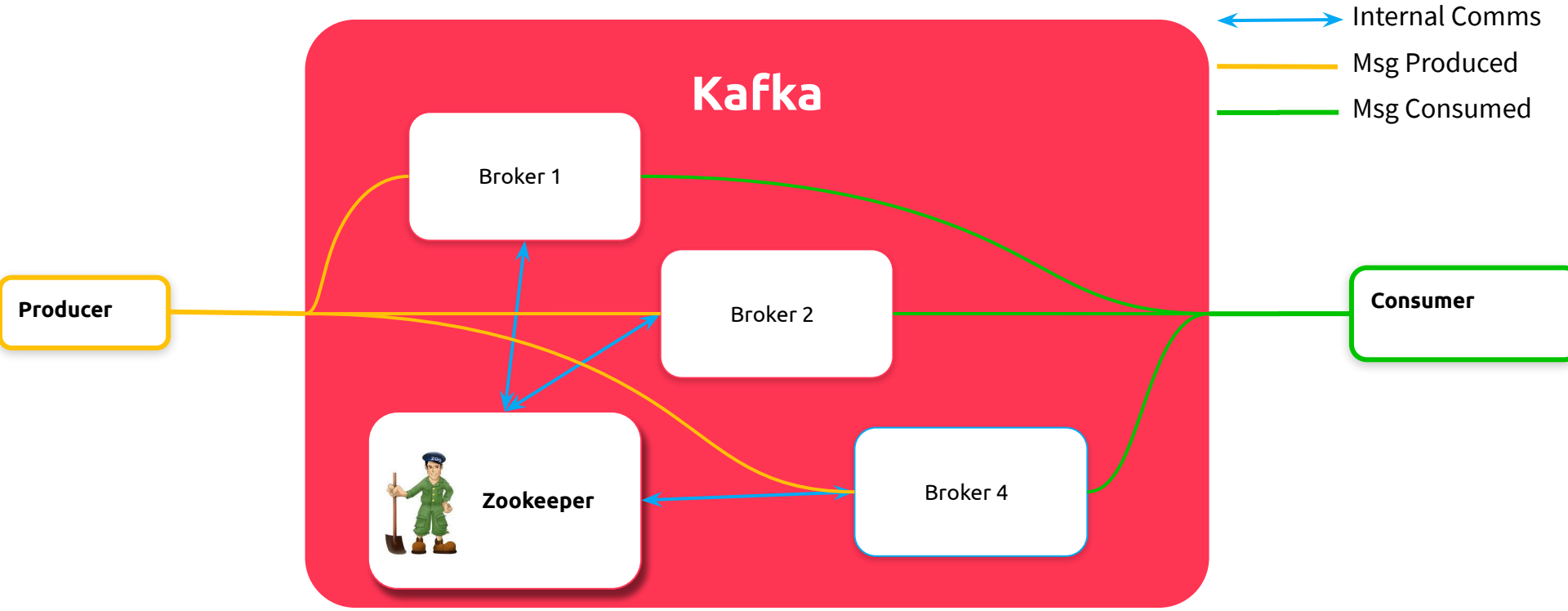
complexity = n + n

Architecture: the WHAT of Kafka

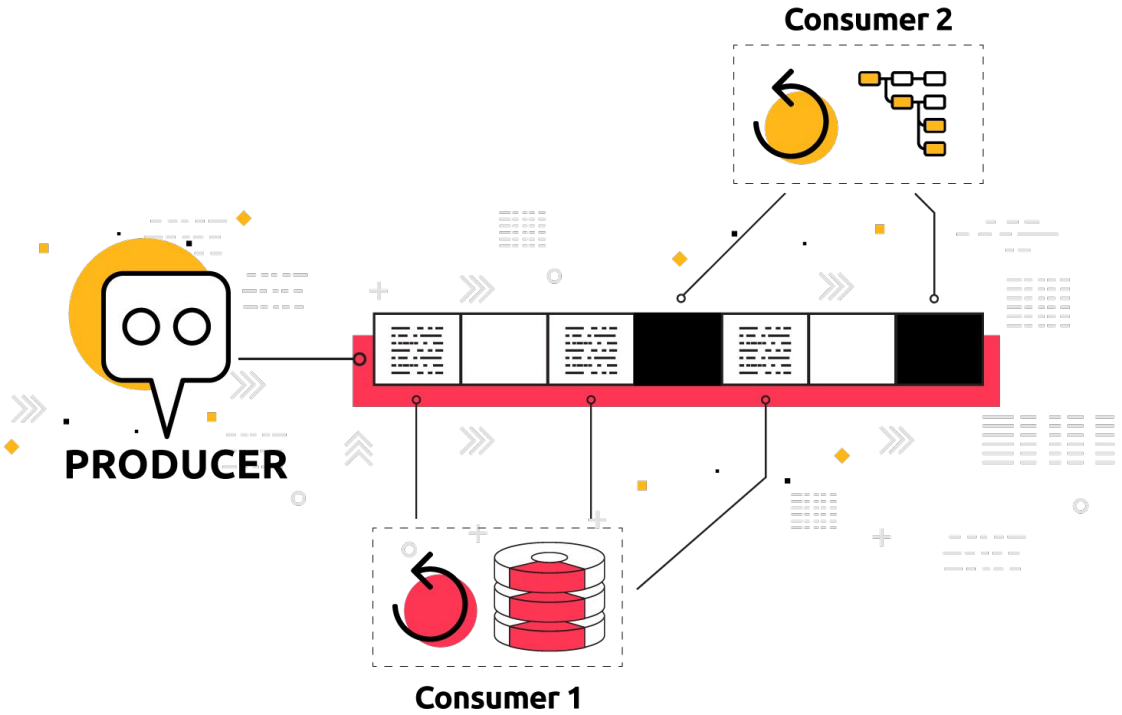
Producin' and Consumin'



Brokers, Clusters, ISRs and Zookeeper

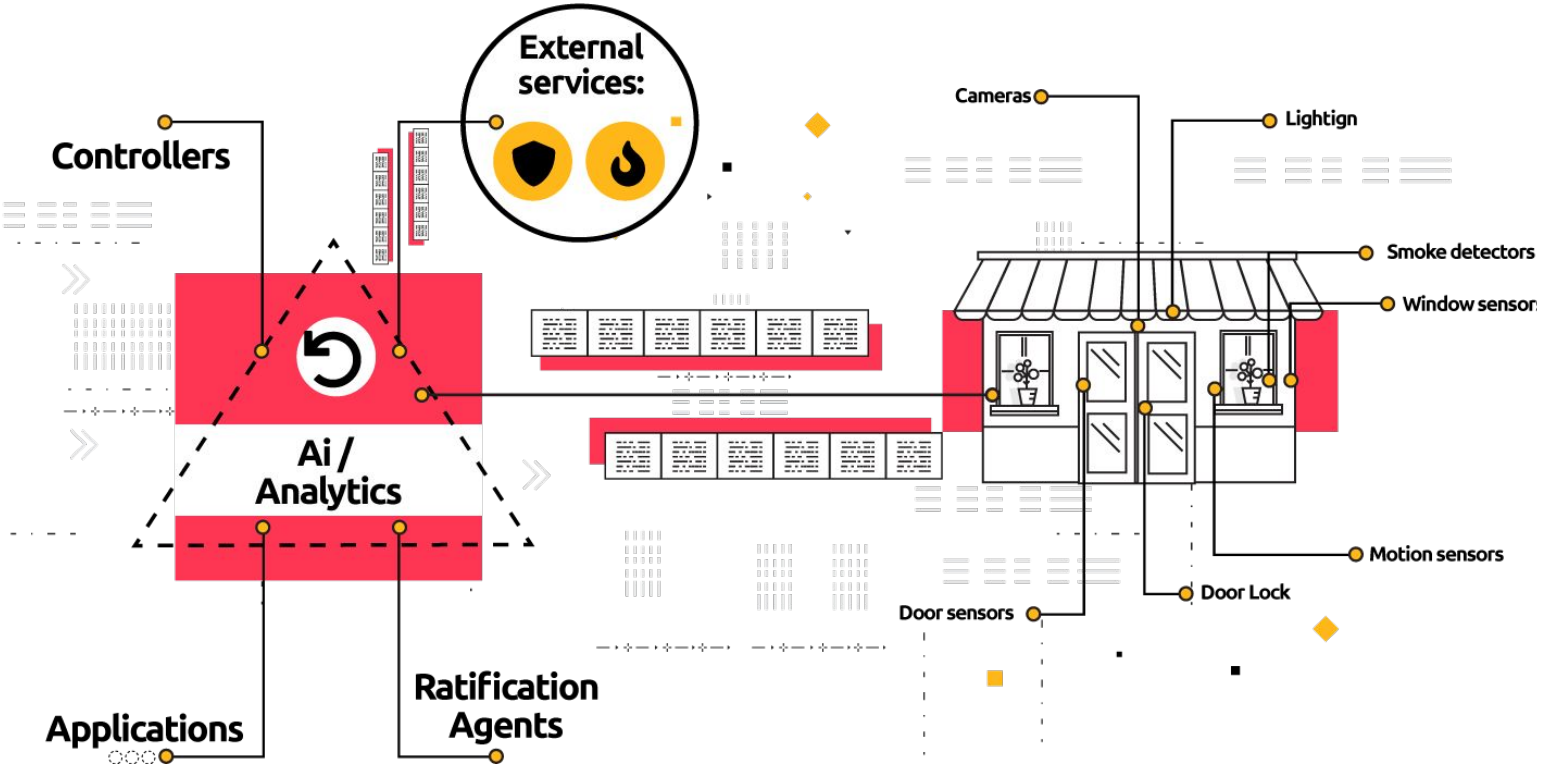


What Apache Kafka does: Publish/Subscribe

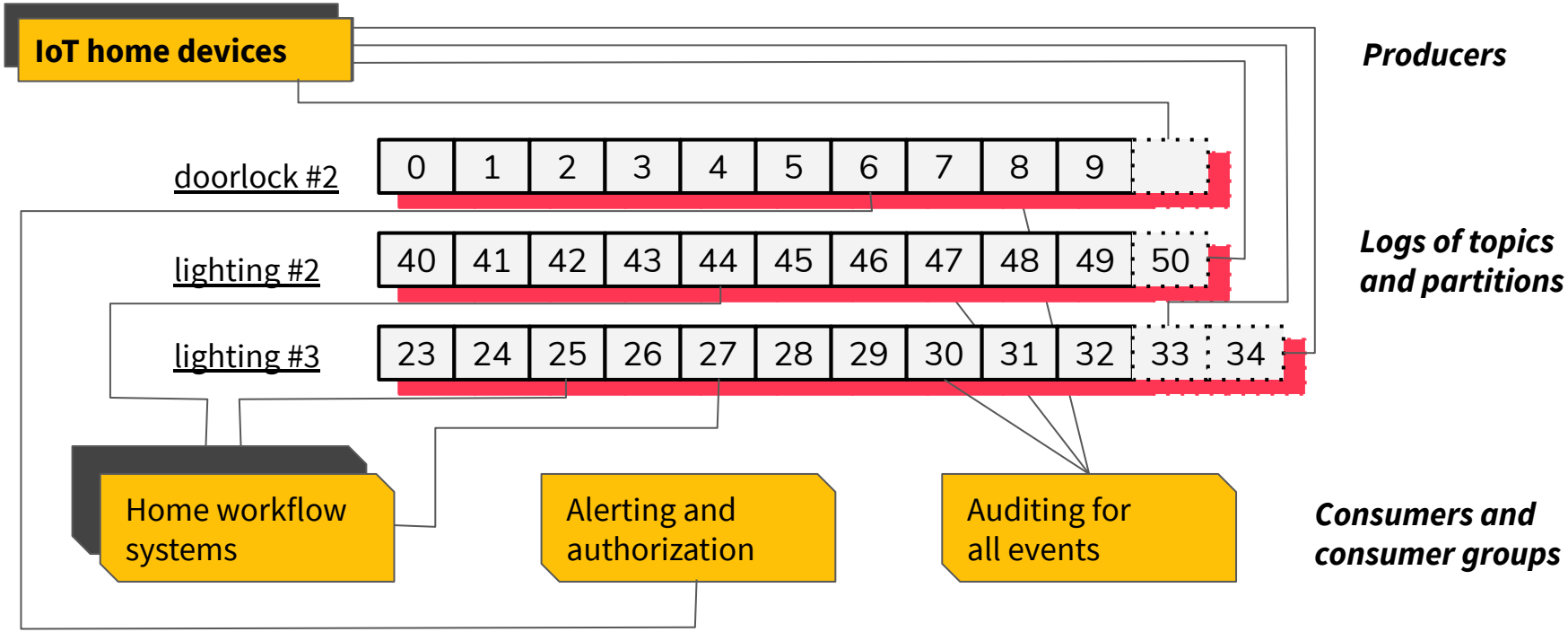


Use-Cases: the HOW of Kafka

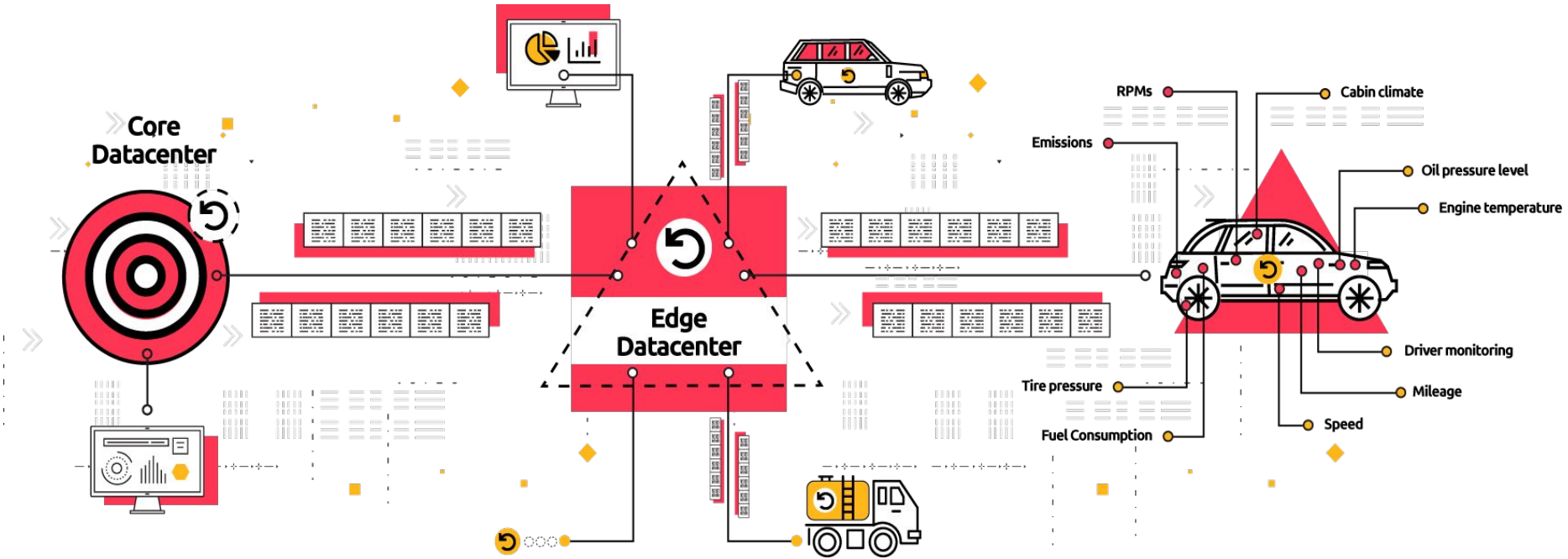
IoT event processing example



How Kafka handles that



Core to Edge example



Advantages of Managed Kafka

WHY Apache Kafka?

- Routing rules configured by consumer
- Built-in support for asynchronous messaging
- High throughput/low latency

Operating Apache Kafka

Covers data pipeline requirements

- Scales to billions of messages per day
- Supports rack and data center aware replication
- Cross-region replication using MirrorMaker
- Real-time streaming
- Decoupling of message consumption & producing
- Client libraries & tools available for all popular languages

But maintenance can be a burden

- Depends on ZooKeeper
- Rebalancing of leaders and partitions
  - In case of failure
  - When scaling up or down
- Broker hangs
- Consider using a managed Kafka service, available from multiple vendors including us

WHY *Managed* Apache Kafka?

- Easy deployment
- No outlay costs
- Thousands of hours of experience
- Auto-scaling
- Centralized, no-fuss management
- High Availability

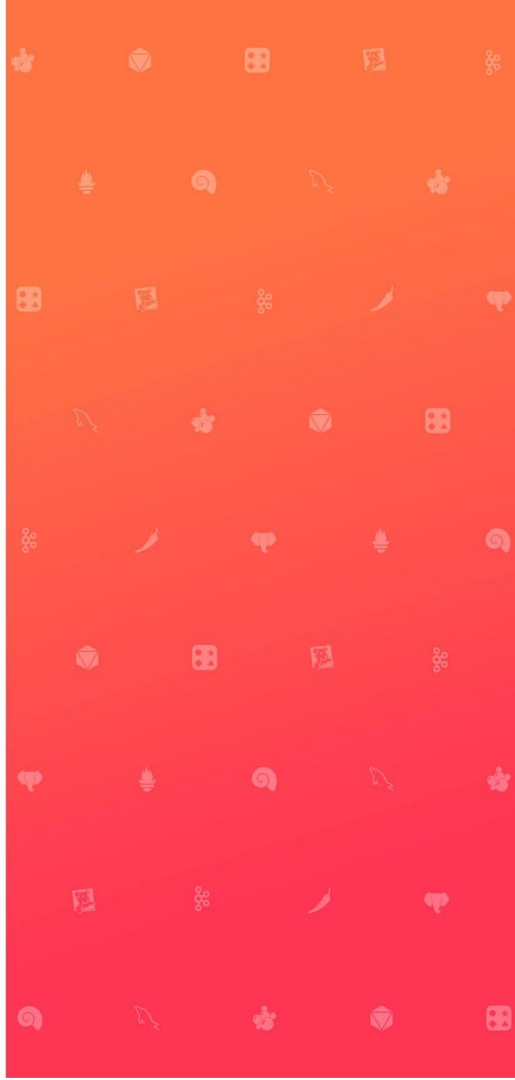
Managed is Peace of Mind

Focus on your product and let us take care of your infrastructure

The
Kafka Ecosystem

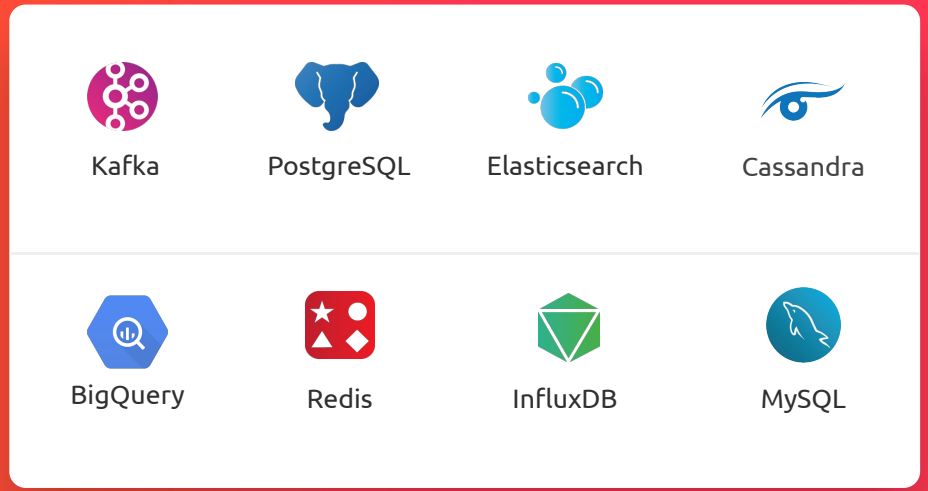
The Kafka Ecosystem

- Kafka Connect
- Schema Registry
- Data Streaming
  - Kafka Streams
  - Spark/Flink



Kafka Connect

- Native Kafka framework for integrating multiple data sources
  - Source Connectors push data to Kafka
  - Sinks dump data
- Runs alongside Kafka OR separately
- Configure with JSON
- Available for: RabbitMQ, Google BigQuery, PostgreSQL and more



Schema Registry

Validate

Ensure message consistency

Validate message source

Many Formats

JSON

Avro

PROTOBUF

Compatibility

Per Schema compatibility

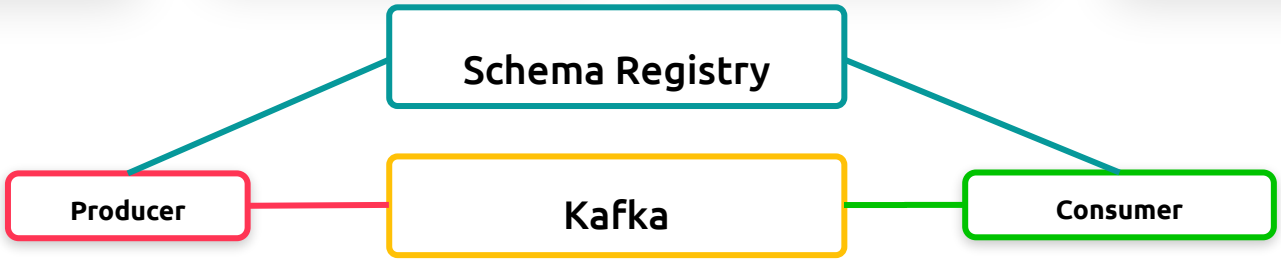
Backwards

Forwards

High support & service level

24/7/365 monitoring by experienced Aiven personnel

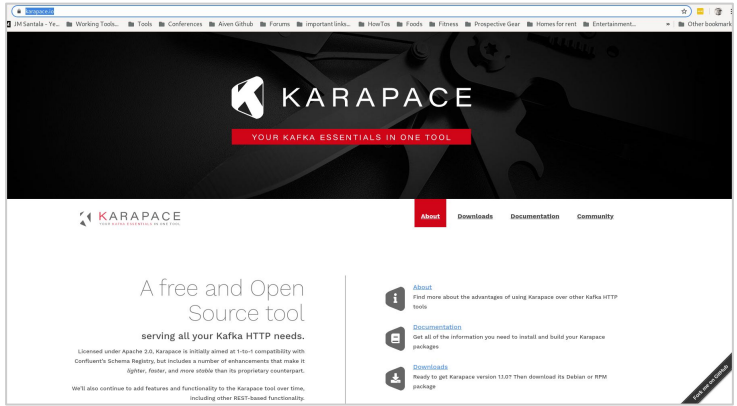
Dedicated phone support available



Karapace

github.com/aiven/karapace

aiven



Schema Registry++

OSI approved license

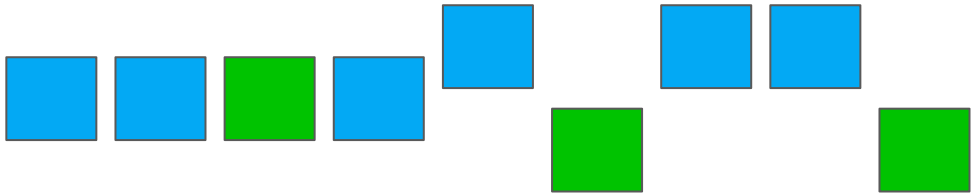
Use with ANY Kafka

Lighter, faster, better, YOURS

Data Streaming -> Flink

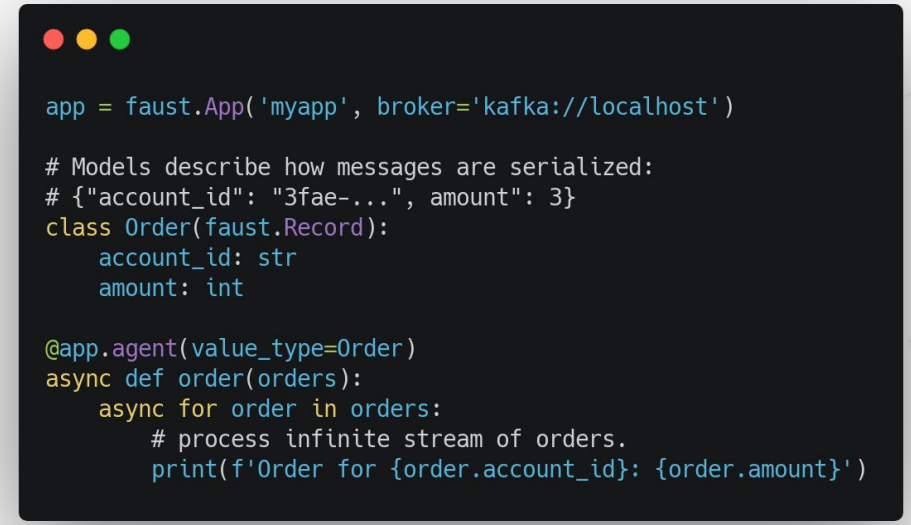
- Stateful Computations over Data Streams
  - Flink lets you run stream and batch processing over any supported data source
- Processing can take place as SQL or as custom code running against the data
- Custom code written in Java or Python

Flink



Kafka Streams

- Stream processing without the need for Spark/Flink
  - Java/Scala native (libraries for many other languages)
- Stateful
- Support out of the box
  - Essentially a Consumer



```python
app = faust.App('myapp', broker='kafka://localhost')

# Models describe how messages are serialized:
# {"account_id": "3fae-...", amount": 3}
class Order(faust.Record):
    account_id: str
    amount: int

@app.agent(value_type=Order)
async def order(orders):
    async for order in orders:
        # process infinite stream of orders.
        print(f'Order for {order.account_id}: {order.amount}')
```

DEMO

Try it yourself!

https://github.com/aiven/presentations/tree/master/Cloud_chats_by_Aiven/Kafka_101

Final words

- **Kafka is:**
  - The evolution of messaging systems
  - The backbone of a Data Pipeline
  - An ecosystem of tools
  - Easy to integrate with popular software
  - Waiting for **you** at **aiven.io**

Produce once, consume everywhere

Thank you for your time!

Q & A

Do You Have Any Questions?

@aiven_io

webinars@aiven.io

Links / Further Reading

- Aiven Kafka
- Karapace
- Aiven's Presentations
- Kafdrop
- Using Kafdrop Web- UI with Aiven Kafka
- Effective Kafka (ebook) (unaffiliated)