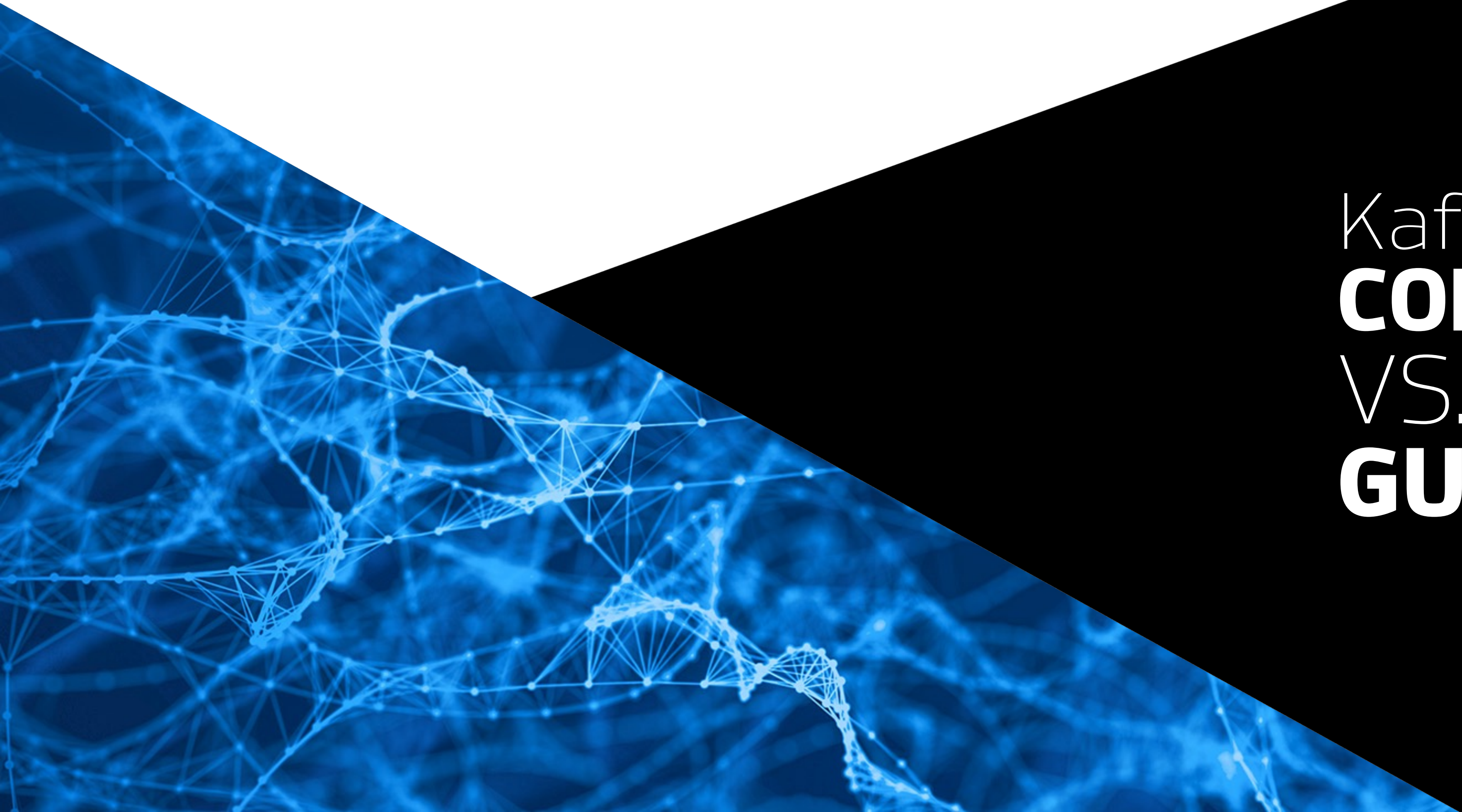
DEEPSHORE

Kafka Basics: **CONSISTENCY** VS. **DELIVERY GUARANTEES**



THERE IS A DIFFERENCE

Let’s understand the difference between

- **Consistency**
- **Delivery Guarantees**

… and how these concepts are related to each other.

01

CONSISTENCY IN ANY IT SYSTEM

ONE WORD – MANY MEANINGS

Centralized systems vs. distributed systems:

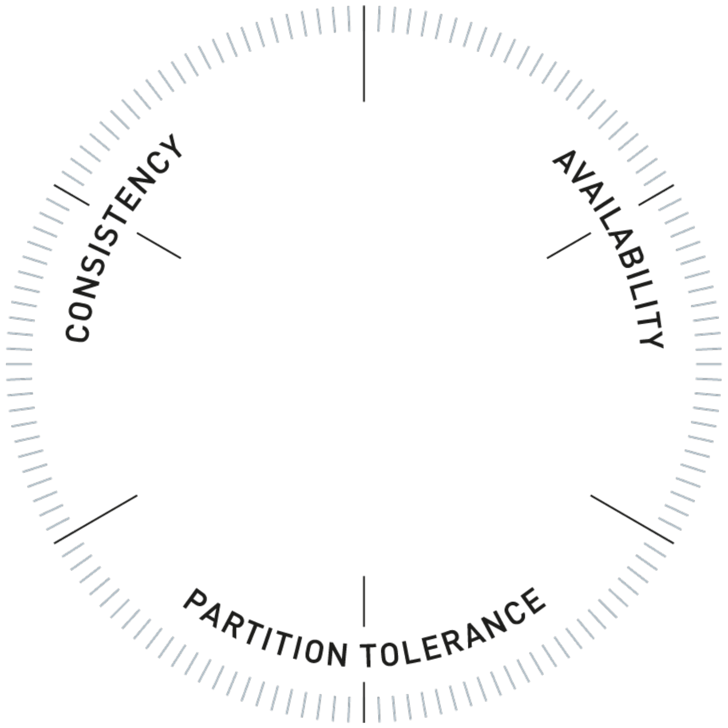
- Two different types of systems with two different approaches
- Using the same word with two different meanings!

CENTRALIZED SYSTEM: ACID TRANSACTIONAL DATABASES

- **Atomic** – guarantees that each transaction is treated as a single „unit“
- **Consistent** – any new transaction to the database won't corrupt the database
- **Isolation** – ensures that concurrent execution of transactions will not interfere
- **Durability** – a transaction which has been committed will remain committed

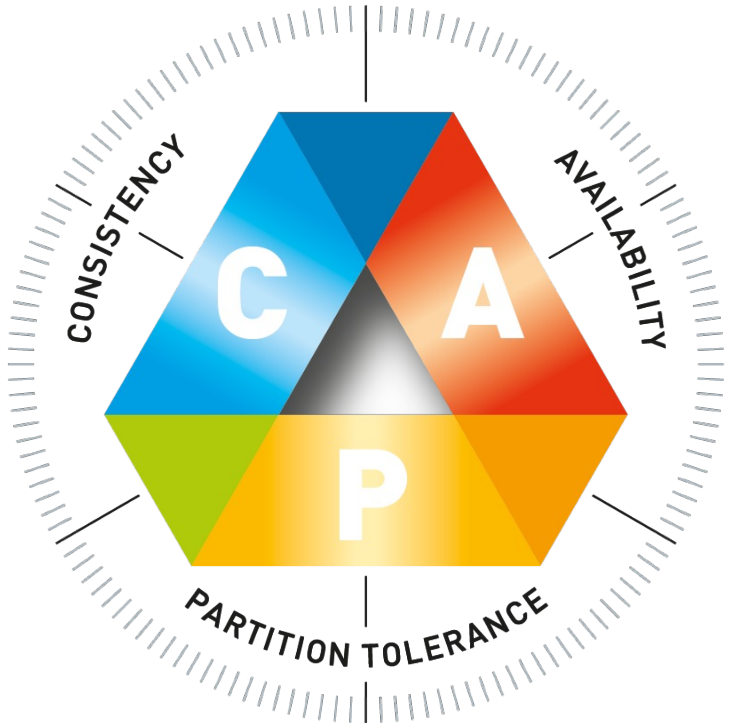
DISTRIBUTED SYSTEM: CAP THEOREM

- **Consistency** – all clients will get the same answer at the same time to the same question
- **Availability** – a client request will get a response from the system at any time
- **Partition tolerance** – the cluster must continue to work despite of a single node failure or a communication breakdown inside the cluster



DISTRIBUTED SYSTEM: CAP THEOREM

The CAP theorem maintains that a distributed system can deliver only two of three desired characteristics.



CONSISTENCY:
ONE WORD, TWO CONCEPTS

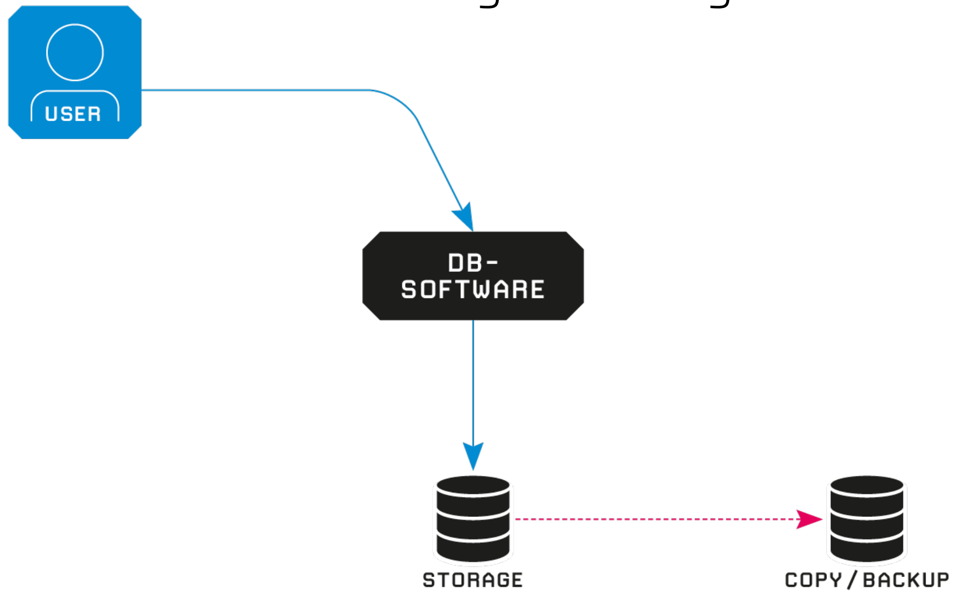
- **ACID** – Consistency describes a characteristic on a transaction level
- **CAP** – Consistency is a general system characteristic
- **Neither concept guarantees information/message delivery!**

KEEP IN MIND

When people are talking about consistency,
they might be talking about different things.

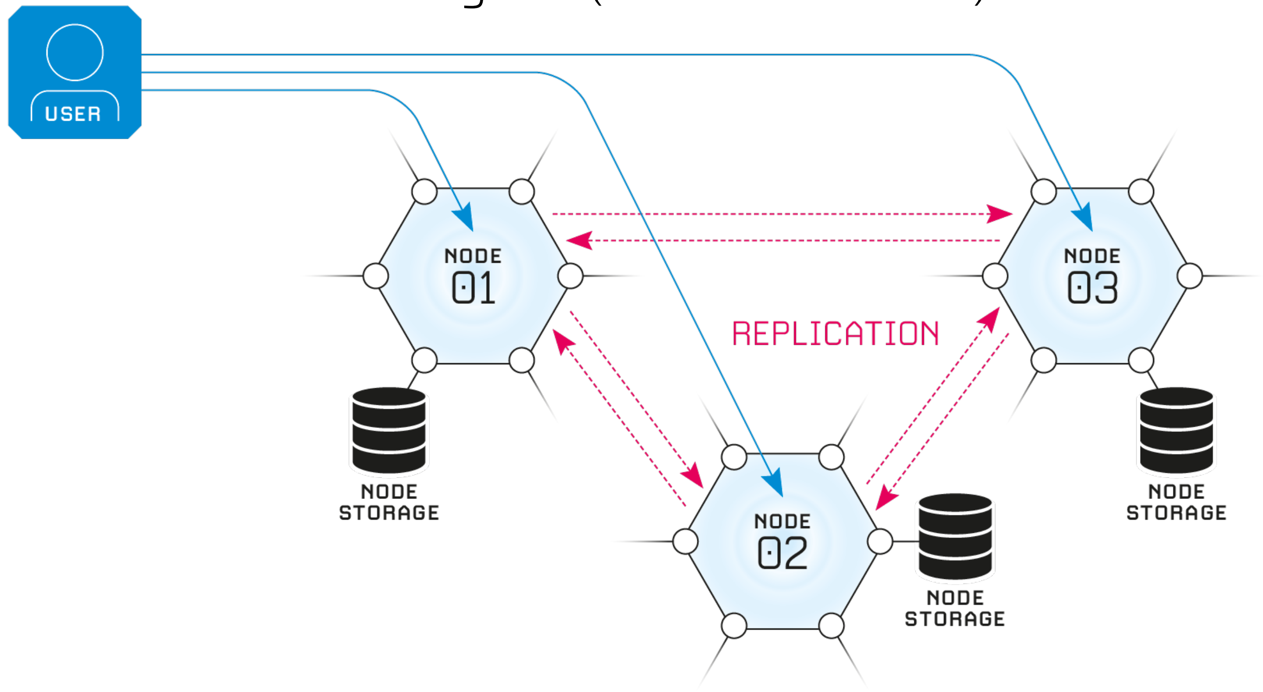
THERE ARE TWO BASIC PATTERNS

RDBMS = avoid any failure or DB inconsistency
→ In case of failures, they can never be 100% available but they will always be consistent (ACID)!



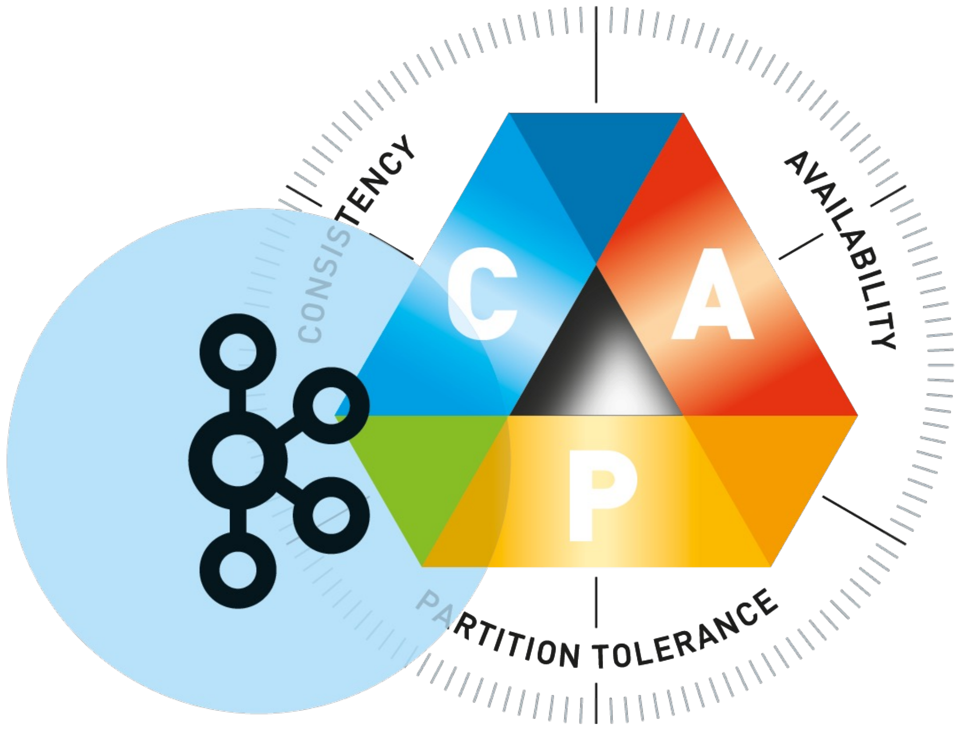
THERE ARE TWO BASIC PATTERNS

Distributed System = failures will happen
and the system has to be able to handle them
→ They can (but don't have to) be consistent



KAFKA AND THE CAP THEOREM

- Kafka is considered to be a distributed system.
- LinkedIn says Kafka fulfills C and A. The fulfillment of A or P depends on the individual set-up.
- Most Kafka systems are more C and P rather than C and A.
- By design, Kafka always fulfills C.



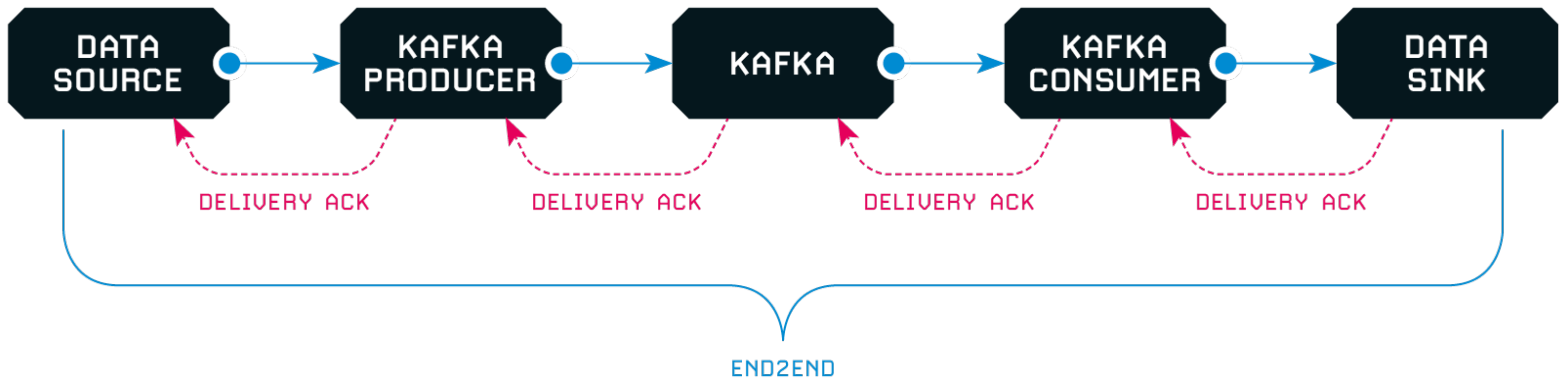
DELIVERY GUARANTEES

MESSAGE DELIVERY

- A message is considered to be delivered when the receiving client acknowledged (ACK) the „write“ back to the sender.
- If a data pipeline consists of more than 2 technical entities, an End2End ACK must always be asynchronous. i.e. a single ACK is not sufficient.
- This also applies to data pipelines with Kafka. It is not possible to solve this problem in Kafka alone, because the End2End ACK will always be asynchronous.

MESSAGE DELIVERY

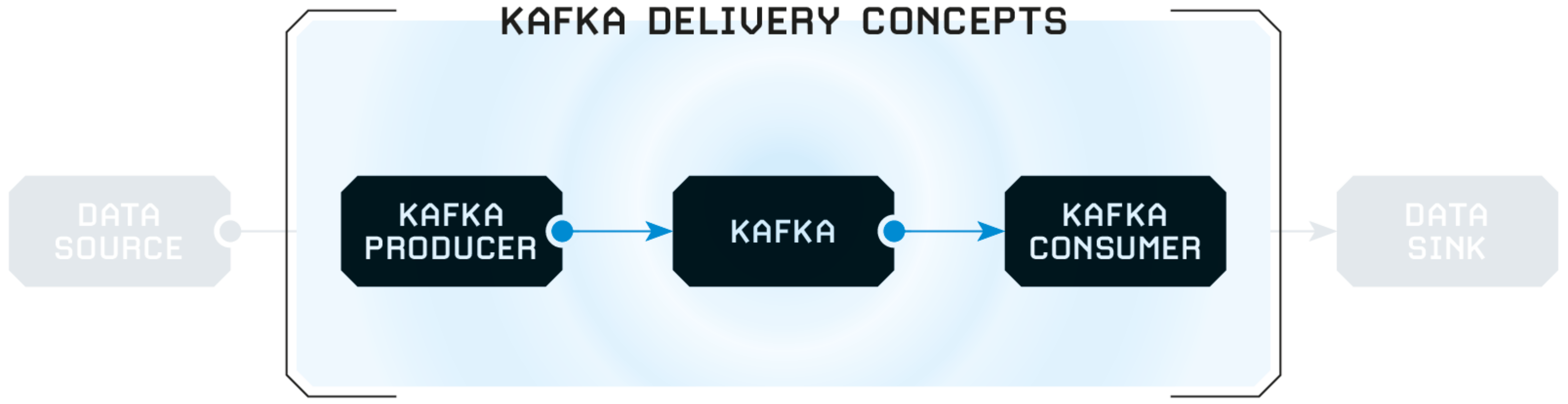
Despite a successful delivery, an End2End consistent data pipeline is not guaranteed.



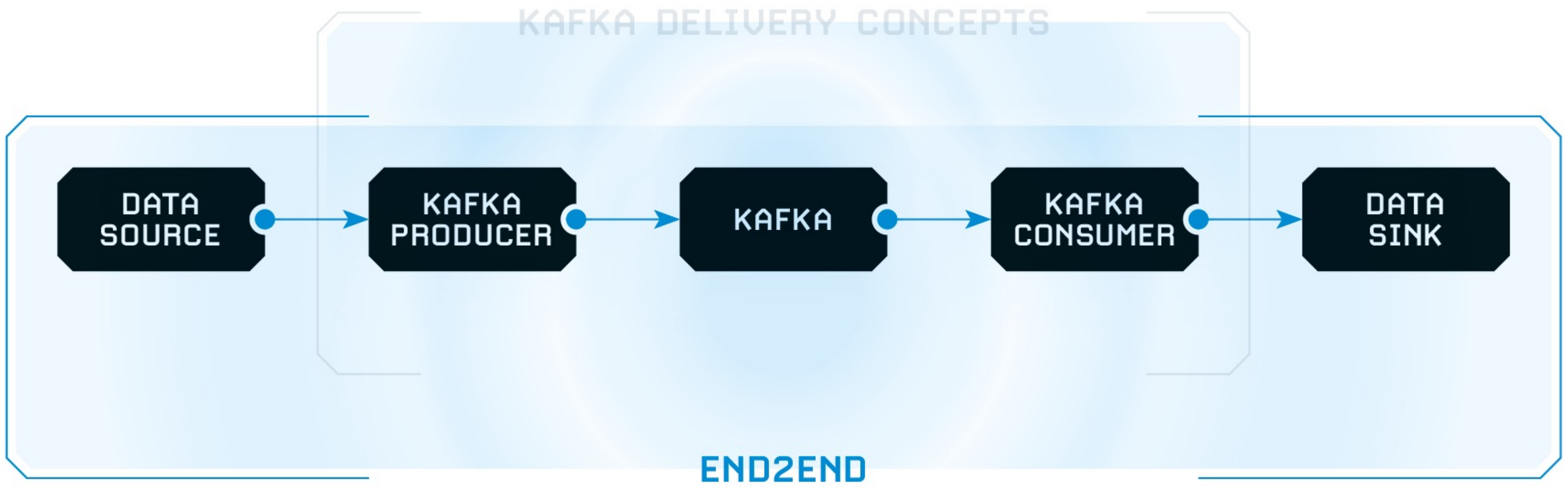
KAFKA'S CONCEPT OF DELIVERY GUARANTEES

- **No Guarantee**
- **At-most once:** Every message is persisted in Kafka at-most-once. Message loss is possible if the producer doesn't retry on failures.
- **At-least-once:** Every message is guaranteed to be persisted in Kafka at-least-once. There is no chance of message loss but the message can be duplicated if the producer retries when the message is already persisted.
- **Exactly-once:** Every message is guaranteed to be persisted in Kafka exactly once without any duplicates and data loss even where there is a broker failure or producer retry.

KAFKA'S CONCEPT OF DELIVERY GUARANTEES: SCOPE



A DATA PIPELINE IS MORE THAN JUST KAFKA



TAKE AWAYS

- Kafka was designed for high throughput but not for a transactional message delivery.
- There is nothing like an *Exactly-Once=true* switch in Kafka.
- Such implementations are quite complex and require a good understanding of general Kafka concepts and system behavior.
- Even if there would be such a simple configuration, it could not guarantee an End2End delivery, since data pipelines are consisting of more than 3 technical entities, where Kafka is just one.
- A transactional like delivery guaranty in a distributed System is a technical antipattern. Because you would force such systems to an ACID like behavior on a per-message level.
- Such configurations will cost performance.

TAKE AWAYS

- If you need a pipeline with guaranteed message delivery and a check of completeness, you could consider to:
  - build a transaction audit-log in a second data pipeline which enables the consumer to check each transaction (e.g. list of Hash Keys)
  - Work with a unique transaction IDs from the source system, to enable the consumer to identify duplicated or missing messages (e.g. foreign ID in the Kafka message header)
  - …not use Kafka

Who said:
KAFKA CONSISTENCY WAS EASY?

CONTACT

Deepshore GmbH · Baumwall 3, D-20459 Hamburg
Telefon +49 40 46664-296 · Fax +49 40 46664-299
E-Mail info@deepshore.de · **www.deepshore.de**