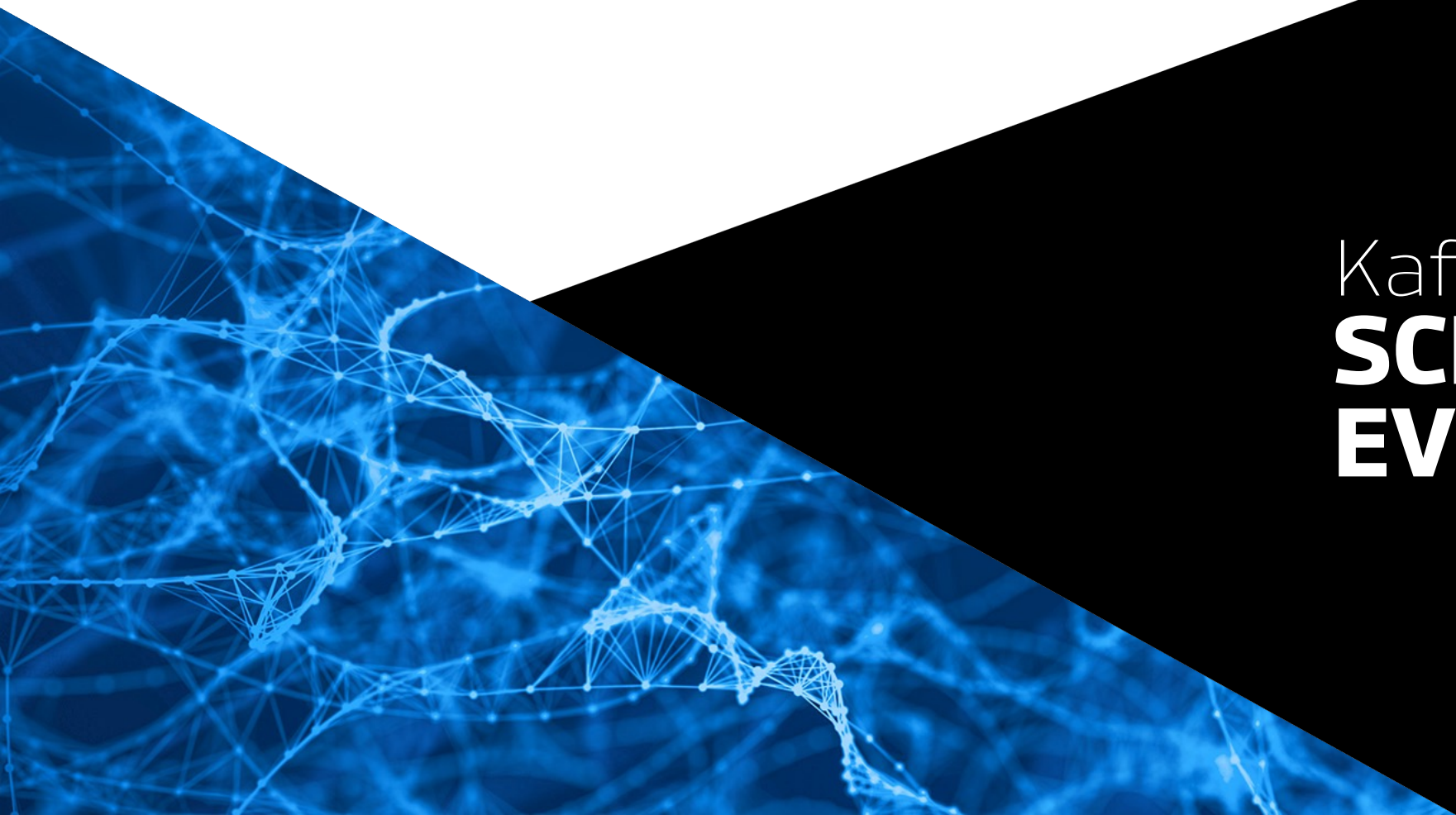
DEEPSHORE



Kafka:
SCHEMA EVOLUTION

DEEPSHORE

The general Kafka design puts a lot of responsibility to its producers, but also to its consumers.

IMPORTANT THINGS TO KNOW ABOUT: SCHEMA EVOLUTION

Let´s understand, ...

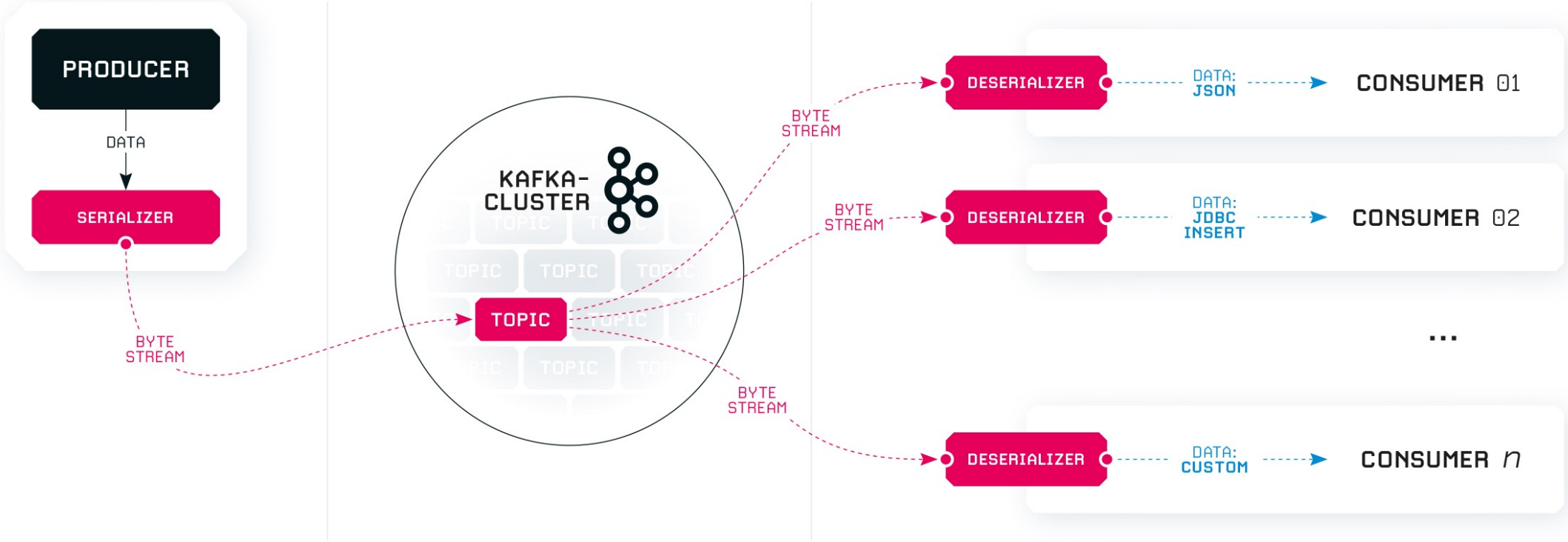
- Why schemas in Kafka?
- How exactly does it work?
- How can schemas be changed?

01

WHY SCHEMAS IN KAFKA?

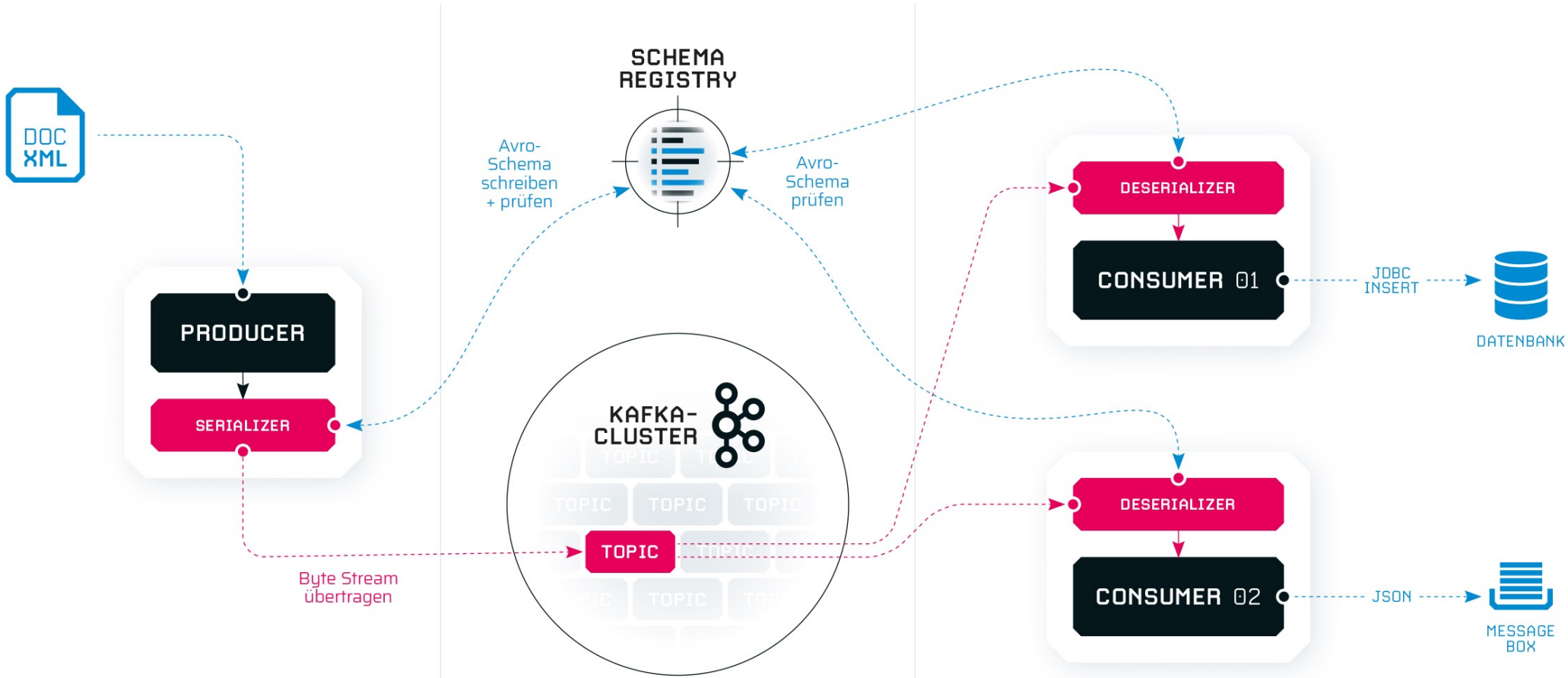
KAFKA ...

- does not care about the data structure of messages since it...
- works with a serialized byte stream – NO STRUCTURE!!!
- Producers and Consumers need to somehow align on: how to serialize/deserialize data and how to interpret such data.
- Without a central Schema, they need to talk to each other.
- Technical producers and consumers must always be compatible!
- Without a central schema, any change in the data structure would lead to a simultaneous change of all producers & consumers; old messages could not be read anymore (or you run old & new versions simultaneously)



KAFKA (CONFLUENT) SCHEMAS...

- are something like a message-data-catalog.
- When well defined, they can explain the structure and business aspects of message data.
- idealy, decouples producer from any consumer.
- are hosted in a central registry.
- is not a Kafka-standard but part of the Confluent distribution!



02

HOW EXACTLY DOES IT WORK?

DEEPSHORE

HOW DOES A PRODUCER SERIALIZE DATA?
HOW DOES A CONSUMER DESERIALIZE DATA?

Format	Producer	Consumer
Avro	io.confluent.kafka.serializers.KafkaAvroSerializer	io.confluent.kafka.serializers.KafkaAvroDeserializer
ProtoBuf	io.confluent.kafka.serializers.protobuf.KafkaProtobufSerializer	io.confluent.kafka.serializers.protobuf.KafkaProtobufDeserializer
JSON Schema	io.confluent.kafka.serializers.json.KafkaJsonSchemaSerializer	io.confluent.kafka.serializers.json.KafkaJsonSchemaDeserializer

THE „WIRE FORMAT"...

- ... is what Confluent-Avro-(de)serilalizer uses under the hood...
- *„Currently supported primitive types are null, Boolean, Integer, Long, Float, Double, String, byte[], and complex type of IndexedRecord. Sending data of other types to KafkaAvroSerializer will cause a SerializationException."* ©Confluent
- *„(...) it is (...) important that the physical byte format of serialized data does not change unexpectedly (...). Even the smallest modification can result in records with the same logical key being routed to different partitions (...)."* ©Confluent

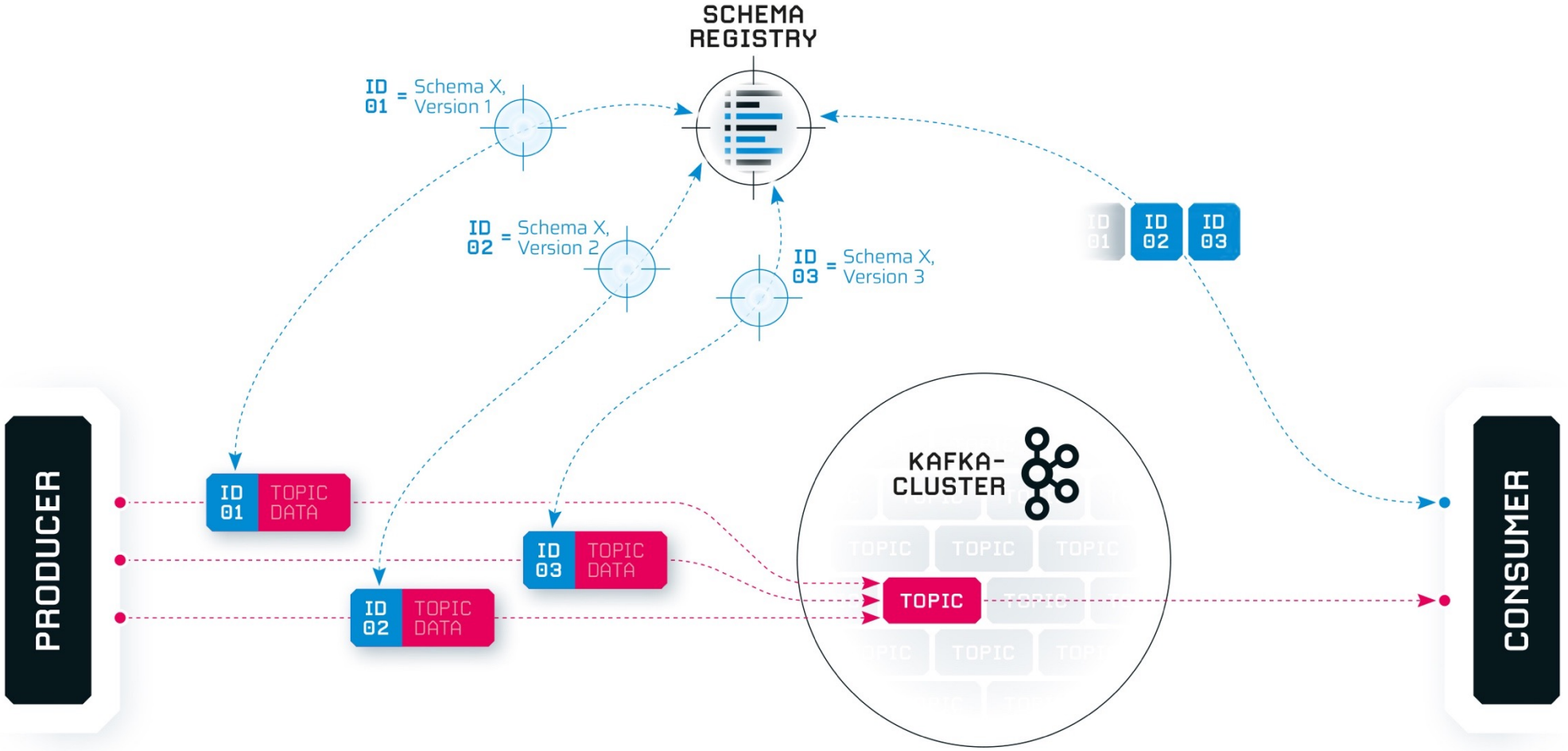
Bytes	Area	Description
0	Magic Byte	Confluent serialization format version number; currently always 0.
1-4	Schema ID	4-byte schema ID as returned by the Schema Registry
5-...	Data	Avro serialized data in Avro's binary encoding. The only exception is raw bytes, which will be written directly without any special Avro

HOW CONFLUENT USES AVRO-(SCHEMAS) ...

- Of course one can use any homemade (de-)serializer, but be carefull... data pipelines might explode due to incompatible plug-ins!

- The moment someone uses KafkaAvroSerializer, the producer needs to connect to a Schema Registry...*mandatory parameter: schema.registry.url*
  (or you build something else like useLocalSchemaFile)

- In general, a SerializationException may occur during the „send call“ from a producer, if the data is not well formed.



03

HOW CAN SCHEMAS BE CHANGED?

THE WORLD WILL MOVE ON – HOW SCHEMAS CAN BE CHANGED ...

- Confluent Kafka supports schema evolution.
- You can work with different versions of the same schema at the same time.
- Do not mix that up with changes in a Topic configuration!
- There is no versioning for Topic configuration.
- Here is what the Confluent distribution delivers out of the box...

Consumer oriented

Compatibility Type	Changes allowed	Check against which schemas	Upgrade first
BACKWARD	•Delete fields •Add optional fields	Last version	Consumers
BACKWARD_TRANSITIVE	•Delete fields •Add optional fields	All previous versions	Consumers
FORWARD	•Add fields •Delete optional fields	Last version	Producers
FORWARD_TRANSITIVE	•Add fields •Delete optional fields	All previous versions	Producers
FULL	•Add optional fields •Delete optional fields	Last version	Any order
FULL_TRANSITIVE	•Add optional fields •Delete optional fields	All previous versions	Any order
NONE	•All changes are accepted	Compatibility checking disabled	Depends

Producer oriented

HOW CONFLUENT USES (AVRO-)SCHEMAS …

- The registry will protect a schema from prohibited changes (according to the chosen compatibility mode).
- But … there is no standard/default gatekeeping process preventing a producer to send schema incompatible data …
- … it is possible to enable a Schema Validation in Confluent-Kafka…
- … but this will cost performance!
- The general Kafka design puts a lot of responsibility to it producers, but also to its consumers.

Who said: **SCHEMA EVOLUTION IS EASY?**

CONTACT

Deepshore GmbH · Van-der-Smissen-Straße 9, 22767 Hamburg
Telefon +49 40 46664-296 · Fax +49 40 46664-299
E-Mail info@deepshore.de · **www.deepshore.de**