Serialization

Paweł Kordek
DATA ENGINEER

@pawel_kordek https://kordek.github.io

So Far

```
KafkaProducer<Integer, String> = new KafkaProducer<>(props);

ProducerRecord<Integer, String> r = new ProducerRecord("topic", "key", "value");

producer.send(r).get();
```

Trip Intents

```
Properties props = new Properties();
props.setProperty("bootstrap.servers", "localhost:9092");
props.setProperty("key.serializer", IntegerSerializer.class.getName());
props.setProperty("value.serializer", StringSerializer.class.getName());

KafkaProducer<Integer, String> = new KafkaProducer<>(props);

ProducerRecord<Integer, String> r = ...
```

Trip Intents

```
Properties props = new Properties();
props.setProperty("bootstrap.servers", "localhost:9092");
props.setProperty("key.serializer", IntegerSerializer.class.getName());
props.setProperty("value.serializer", StringSerializer.class.getName());

KafkaProducer<Integer, TripIntent> = new KafkaProducer<>(props);

ProducerRecord<Integer, TripIntent> r = ...
```

Trip Intents

```
Properties props = new Properties();
props.setProperty("bootstrap.servers", "localhost:9092");
props.setProperty("key.serializer", IntegerSerializer.class.getName());
props.setProperty("value.serializer",   );

KafkaProducer<Integer, TripIntent> = new KafkaProducer<>(props);

ProducerRecord<Integer, TripIntent> r = ...
```

StringSerializer won't work anymore

Widely-known

Human-readable

Supported by lots of languages

It Is Not Compact

```
{
   "userId": 83290,
   "latLonFrom": "36.0862,-115.1503",
   "latLonTo": "36.1684,-115.2908"
}
```

```
'{"userId": 83290,"latLonFrom": "36.0862,-
115.1503","latLonTo": "36.1684,-115.2908"}'
```

Changing Format Is Tricky

```
{
    "userId": 83290,
    "latLonFrom": "36.0862,-115.1503",
    "latLonTo": "36.1684,-115.2908"
}
```

Changing Format Is Tricky

```
{
    "userId": 83290,
    "latLonFrom": "36.0862,-115.1503",
    "latFrom": 36.0862,
    "lonFrom": -115.1503,
    "latLonTo": "36.1684,-115.2908",
    "latTo": 36.1684,
    "lonTo": -115.2908,
}
```

Become deprecated

Serialization/deserialization cost is not negligible

Desired Format

- Compact
- Fast
- Language-agnostic
- Easy schema evolution

Serialization Frameworks

Binary Format

```
{
    "userId": 83290,
    "latLonFrom": "36.0862,-115.1503",
    "latLonTo": "36.1684,-115.2908"
}
```

```
"\b\364\301\a\022\01748.2026,16.3688\032\01748.1869,16.3133"
```

Popular Ones

Protocol Buffers

Apache Avro

Protobuf Schema Definition

trip_intent.proto

```
message TripIntent {
    required int32 user_id = 1;
    required string lat_lon_from = 2;
    required string lat_lon_to = 3;
}
```

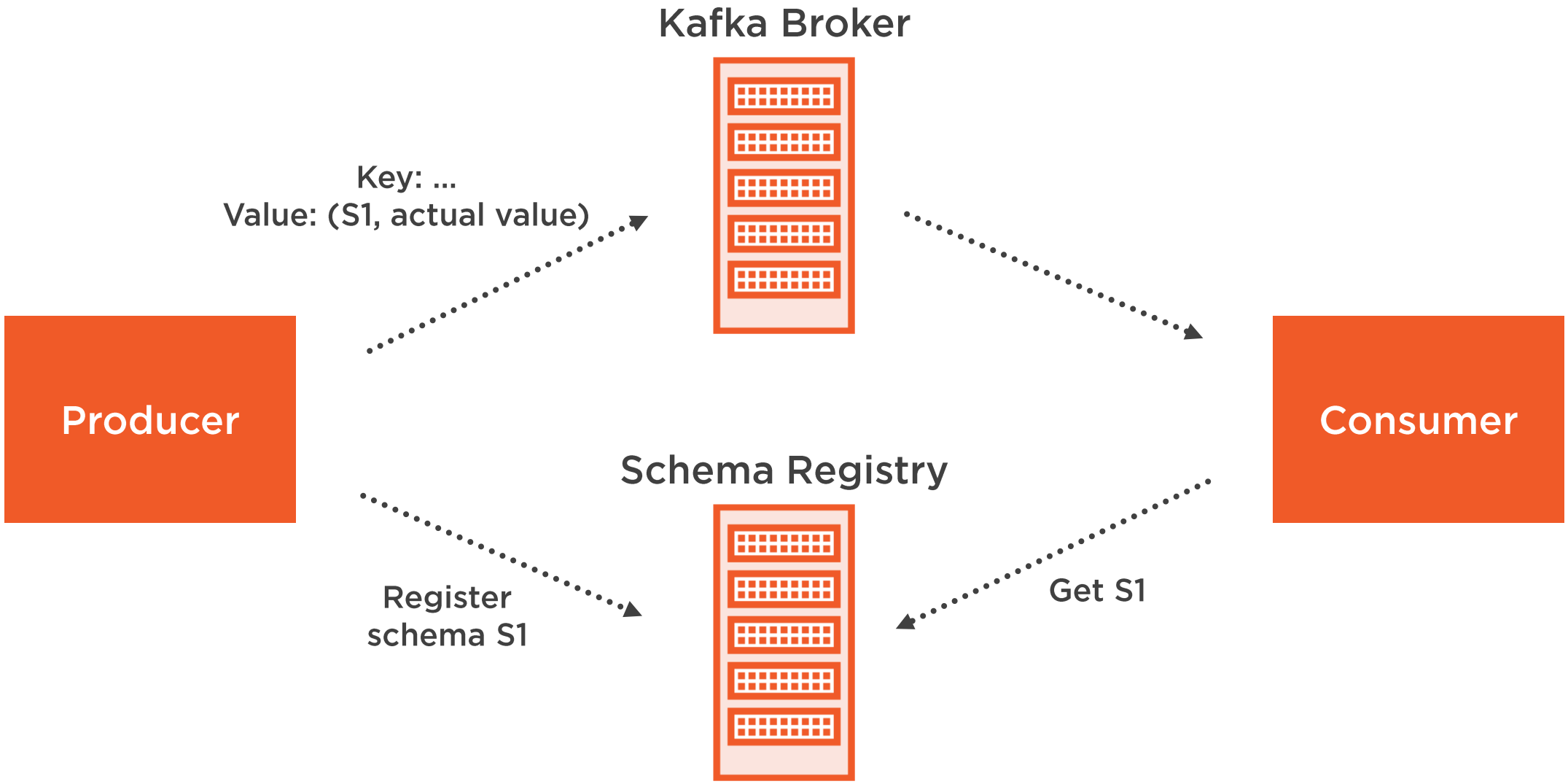
```
TripIntent intent = new TripIntent.newBuilder()
    .setUserId(...)
    .setLatLonFrom(...)
    .setLatLonTo(...)
    .build()

var latLonFrom = intent.getLatLonFrom();
```

Compiled to Java

Using protoc compiler

Schema Registry


Kafka Broker
Key: ...
Value: (S1, actual value)
Producer
Consumer
Schema Registry
Register
schema S1
Get S1

pickup_drive.proto

```
message PickupDrive {
    required int32 driver_id = 1;
    required string lat_lon_from = 2;
    required string lat_lon_to = 3;
}
```

Dynamic Message Handling Has Some Performance Cost

Summary

Binary formats improve efficiency and simplify schema management

Avro is similar to Protobuf, but always requires schema registry

Understanding how message schemas are handled is essential