

# Communicating Messages Structure with AVRO and Schema Registry

---



**Bogdan Sucaciu**

SOFTWARE ENGINEER @ AXUAL

@BSucaciu



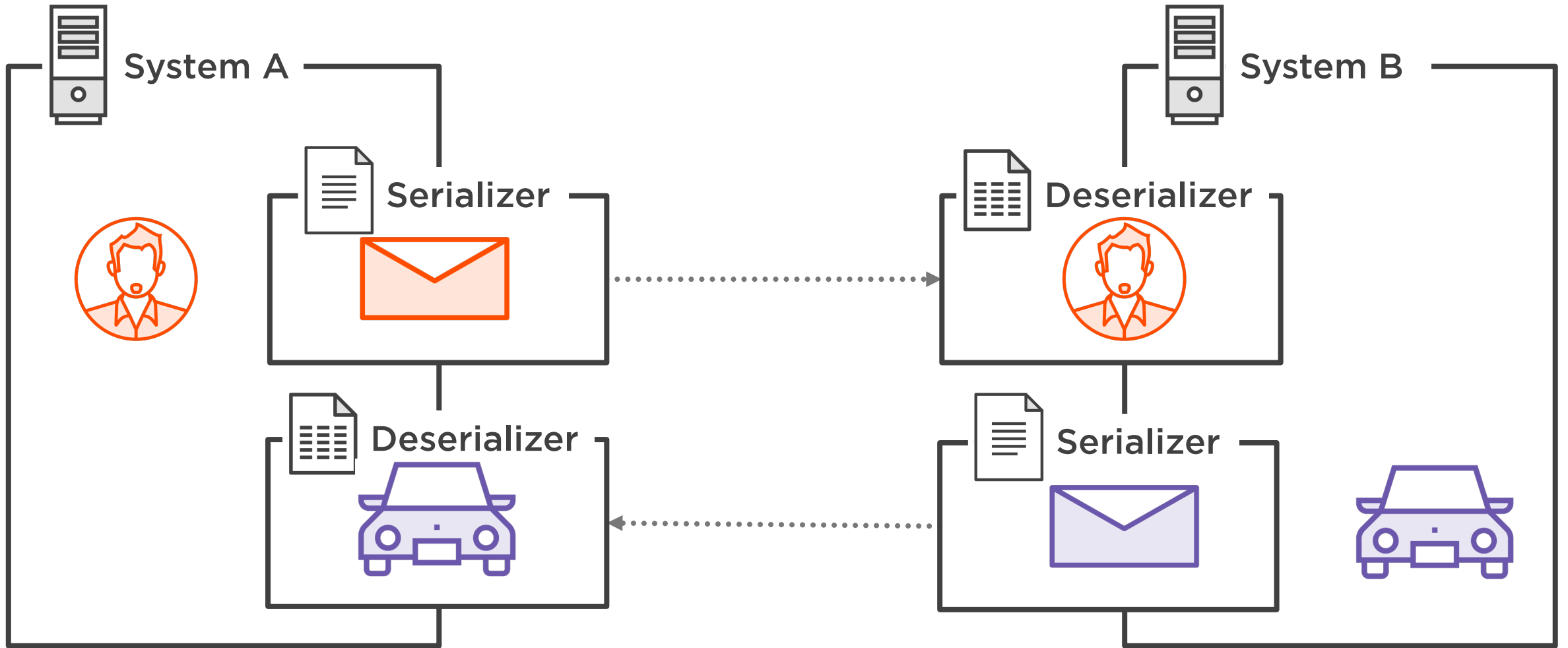
# Serialization

The process of translating **data structures** or **object states** into a format that can be **stored** or **transmitted** and reconstructed later, possibly in a **different** computer environment.

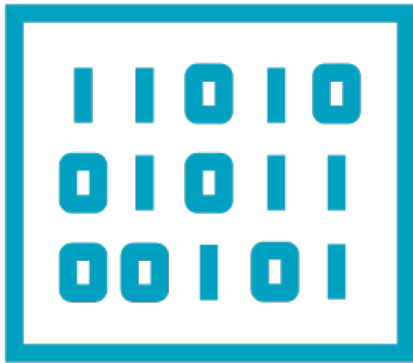
-Wikipedia



# Serialization



# Serialization Formats



## Binary ?

Data is transmitted/stored as bytes or plaintext















## Schema ?

Fixed data structure or flexible



# Serialization Formats

Name	Binary	Schema - IDL
JSON		
XML		
YAML		
AVRO		
Protobuf		
Thrift		



# AVRO



Binary



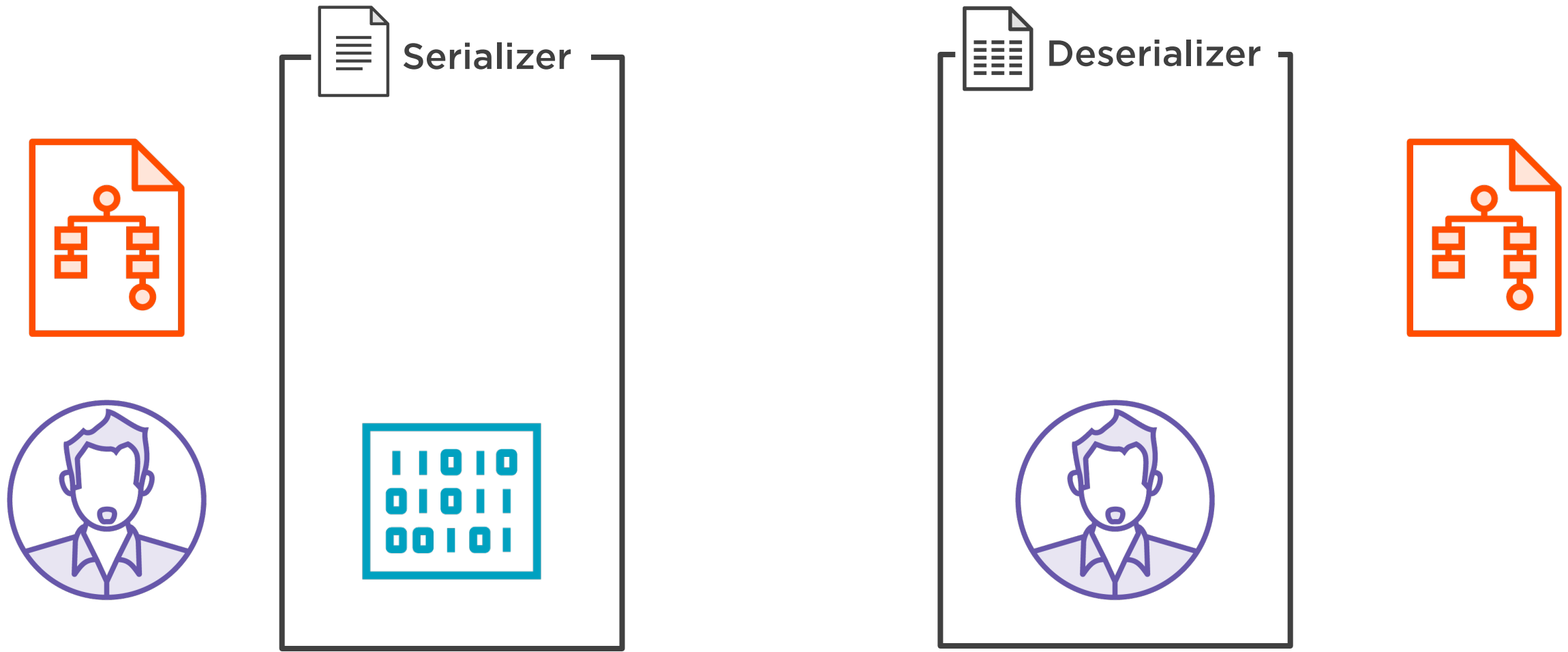
Not Human Readable



Schema



# AVRO Serialization / Deserialization



```
user_schema.avsc
```

```
{  
  "type": "record",  
  "namespace": "com.pluralsight",  
  "name": "User",  
  "fields": [  
    {  
      "name": "userId",  
      "type": "string"  
    }, {  
      "name": "username",  
      "type": "string"  
    }, {  
      "name": "dateOfBirth",  
      "type": "int",  
      "logicalType": "date"  
    }  
  ]  
}
```

## ◀ JSON

- ◀ Defines a complex type
- ◀ Prefix the full name
- ◀ Name of the schema
- ◀ Declaring fields contained by the record
- ◀ e.g: "ABC123"

## ◀ Plain string

## ◀ Special type





# AVRO Types

## Primitives

**null:** no value

**boolean:** a binary value

**int:** 32-bit signed integer

**long:** 64-bit signed integer

**float:** (32-bit) floating-point number

**double:** (64-bit) floating-point number

**bytes:** sequence of 8-bit unsigned bytes

**string:** unicode character sequence

## Complex

records

enums - { "symbols" : [ "BLUE", "GREEN" ] }

arrays

maps - { "values" : "long" }

unions - { [ "null", "string" ] }

fixed - { "size" : "6" }



# Demo

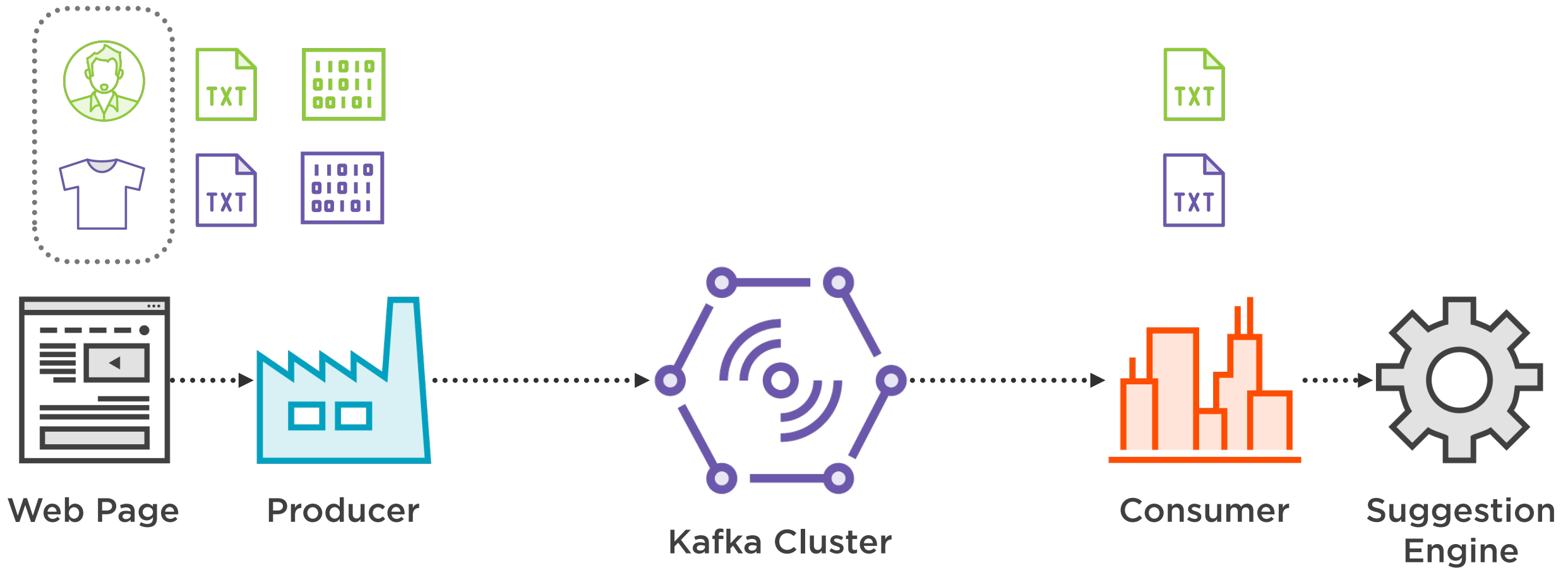


Create AVRO user schema

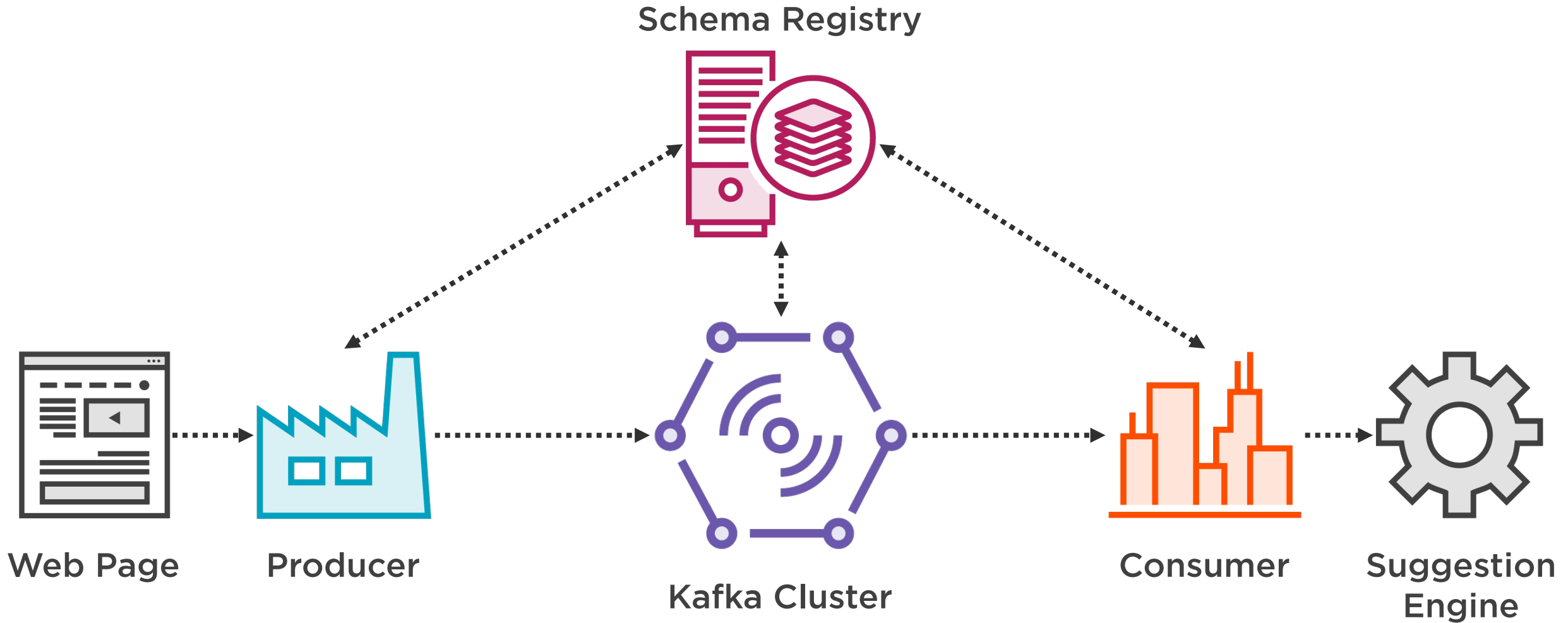
Generate JAVA classes



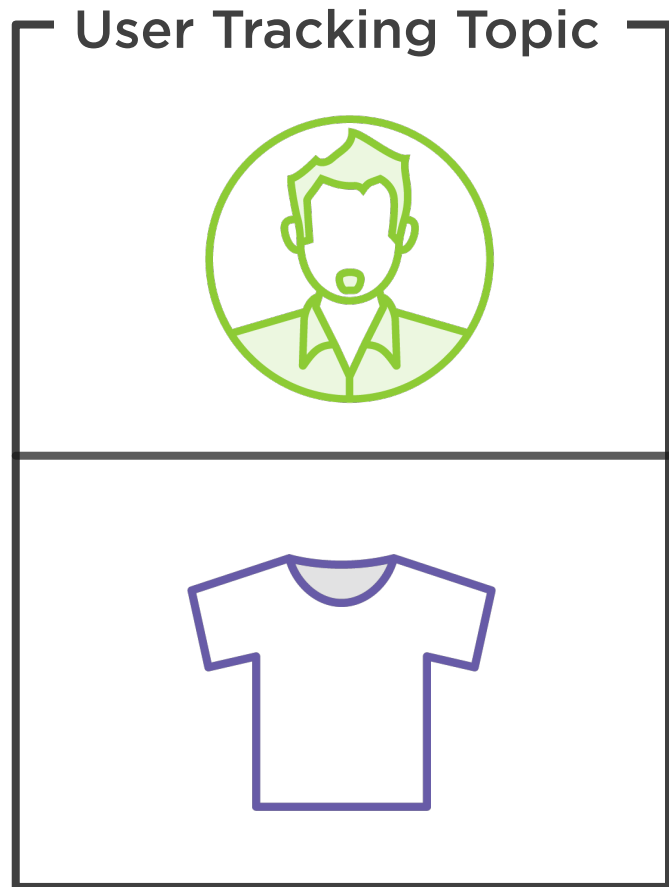
# Applications Architecture



# Applications Architecture



# Subject Name Strategy

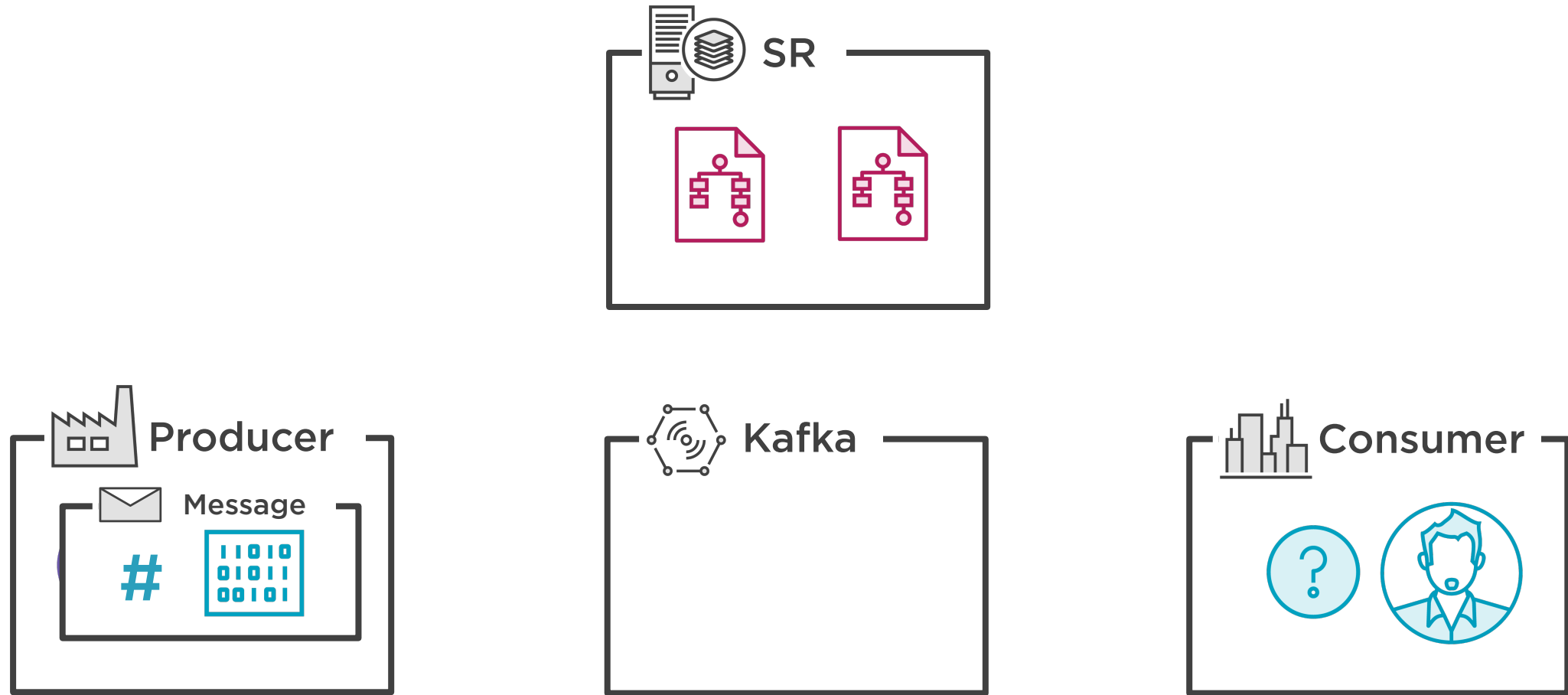


`{topic-name}-key: user-tracking-key`

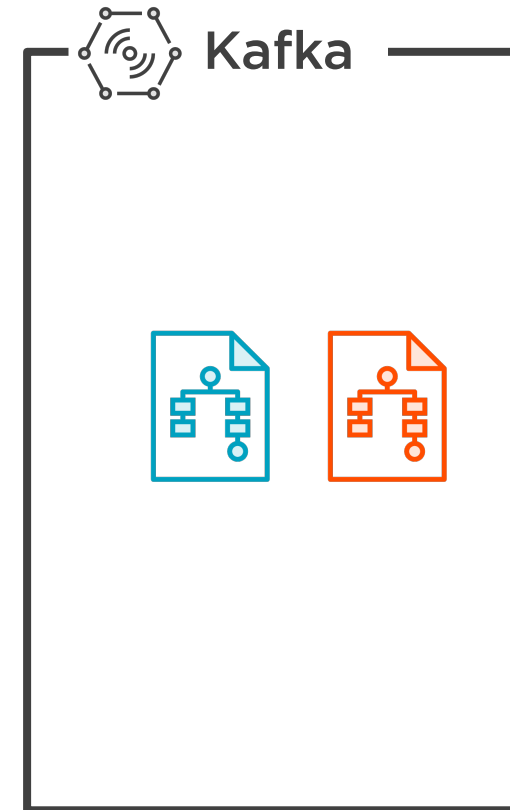
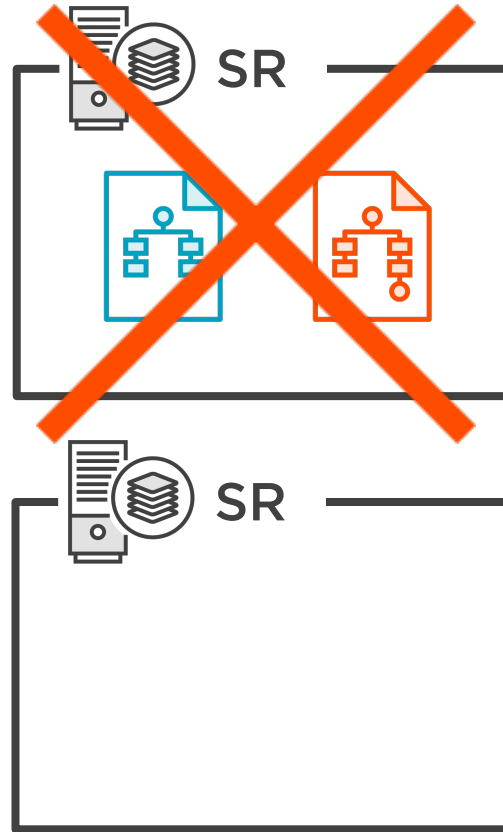
`{topic-name}-value: user-tracking-value`



# Schema Distribution



# Schema Upload



# Demo



## Set up Schema Registry

## Upgrade user tracking applications

- Producer
- Consumer





# Confluent Community License

You can **access** the source code and **modify** or **redistribute** it; there is only one thing you cannot do, and that is use it to make a competing SaaS offering.

-Confluent Website



# Summary



**Serialization formats**

**AVRO to the rescue**

**Schema distribution using Schema Registry**

