aiven

Boost Apache Kafka with Schema Registry

Francesco Tisiot - Senior Developer Advocate

@ftisiot

Apache Kafka doesn't care

```
{
    "id": 778,
    "shop": "Luigis Pizza",
    "name": "Edward Olson",
    "phoneNumbers":
        ["(935)503-3765x4154","(935)12345"],
    "address": "Unit 9398 Box 2056\nDPO AP 24022",
    "image": null,
    "pizzas": [
        {
            "pizzaName": "Salami",
            "additionalToppings": ["🥓", "🌶"]
        },
        {
            "pizzaName": "Margherita",
            "additionalToppings": ["🍌", "🌶", "🍍"]
        }
    ]
}
```


Carlo, Margherita, 2
Francesco, Salami, 1


```
dGVzdCx0ZXN0MSx0ZXN0Mix0ZXN0LHRlc3QxLHRlc3Q
yLHRlc3QsdGVzdDEsdGVzdDIs
```

The Beginning

JSON


```
{
    "id": 778,
    "shop": "Luigis Pizza",
    "name": "Edward Olson",
    "phoneNumbers":
        ["(935)503-3765x4154","(935)12345"],
    "address": "Unit 9398 Box 2056\nDPO AP 24022",
    "image": null,
    "pizzas": [
        {
            "pizzaName": "Salami",
            "additionalToppings": ["🥓", "🌶"]
        },
        {
            "pizzaName": "Margherita",
            "additionalToppings": ["🍌", "🌶", "🍍"]
        }
    ]
}
```

A bit Heavy?

JSON


```
{
    "id": 778,
    "shop": "Luigis Pizza",
    "name": "Edward Olson",
    "phoneNumbers":
        ["(935)503-3765x4154","(935)12345"],
    "address": "Unit 9398 Box 2056\nDPO AP 24022",
    "image": null,
    "pizzas": [
        {
            "pizzaName": "Salami",
            "additionalToppings": ["🥓", "🌶"]
        },
        {
            "pizzaName": "Margherita",
            "additionalToppings": ["🍌", "🌶", "🍍"]
        }
    ]
}
```

Schema

```
{
    "id": int,
    "shop": string,
    "name": string,
    "phoneNumbers":
        [string],
    "address": string,
    "image": string,
    "pizzas": [
        {
            "pizzaName": string,
            "additionalToppings": [string]
        }]
}
```

Payload

```
{
    778,
    "Luigis Pizza",
    "Edward Olson",
    ["(935)503-3765x4154","(935)12345"],
    "Unit 9398 Box 2056\nDPO AP 24022",
    null,
    [
        {
            "Salami",
            ["🥓", "🌶"]
        },
        {
            "Margherita",
            ["🍌", "🌶", "🍍"]
        }
    ]
}
```

```
{
    "id": 778,
    "shop": "Luigis Pizza",
    "name": "Edward Olson",
    "phoneNumbers":
        ["(935)503-3765x4154","(935)12345"],
    "address": "Unit 9398 Box 2056\nDPO AP 24022",
    "image": null,
    "pizzas": [
        {
            "pizzaName": "Salami",
            "additionalToppings": ["🥓", "🌶"]
        },
        {
            "pizzaName": "Margherita",
            "additionalToppings": ["🍌", "🌶", "🍍"]
        }
    ]
}
```

```
{
    778,
    "Luigis Pizza",
    "Edward Olson",
    ["(935)503-3765x4154","(935)12345"],
    "Unit 9398 Box 2056\nDPO AP 24022",
    null,
    [
        {
            "Salami",
            ["🥓", "🌶"]
        },
        {
            "Margherita",
            ["🍌", "🌶", "🍍"]
        }
    ]
}
```

Benefits

Less data in transit

Less data stored

Network is a bottleneck

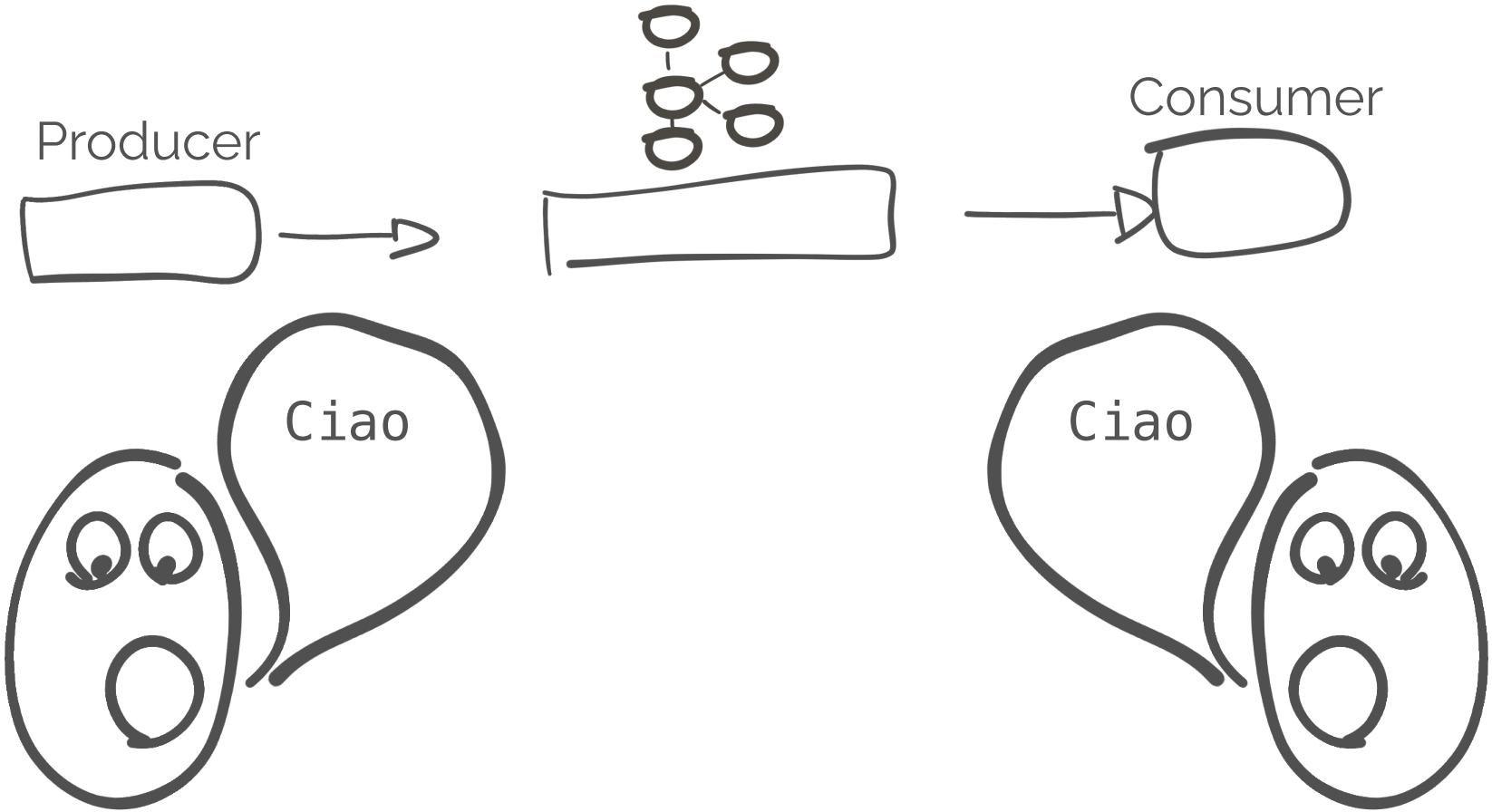
better performance

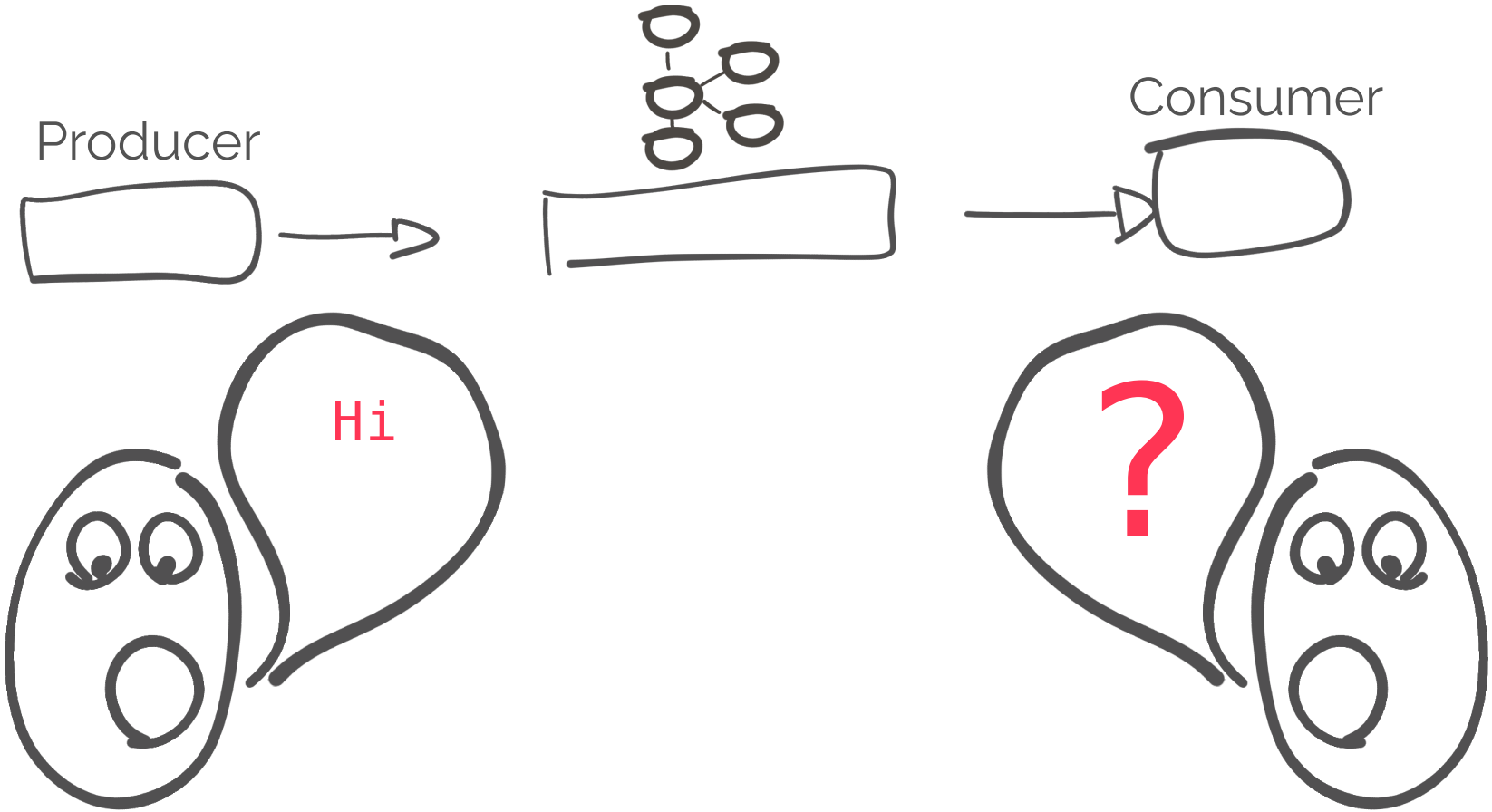
Cons

You need to translate

Is it only about Compression?

No input validation


Producer
Consumer
Ciao
Ciao


Producer
Consumer
Hi
?

JSON

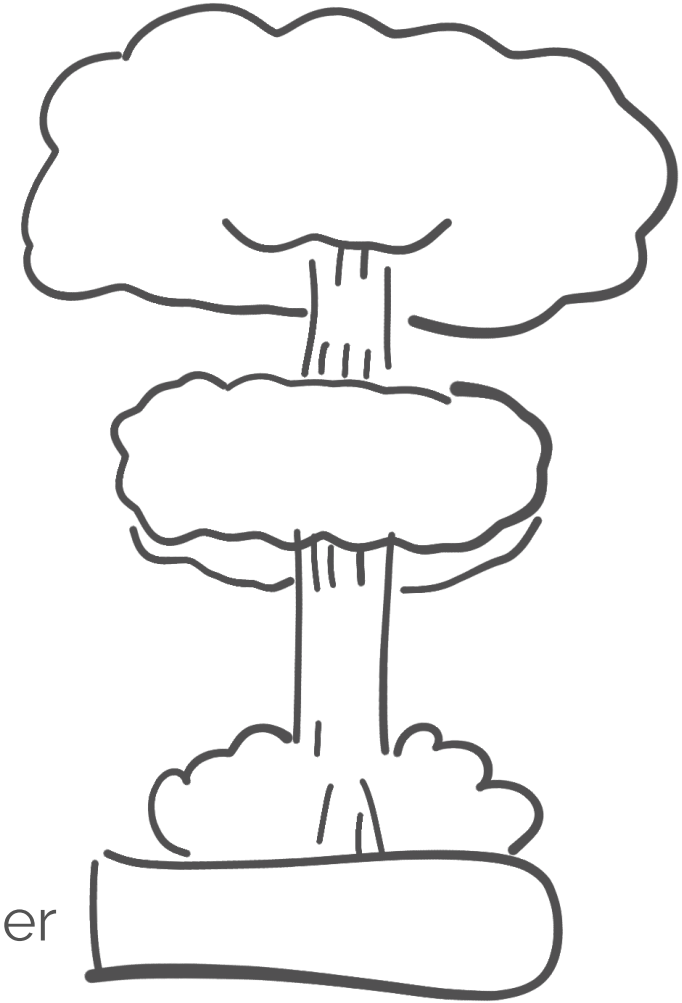

```
{
    "id": 778,
    "shop": "Luigis Pizza",
    "name": "Edward Olson",
    "phoneNumbers":
        ["(935)503-3765x4154","(935)12345"],
    "address": "Unit 9398 Box 2056\nDPO AP 24022",
    "image": null,
    "pizzas": [
        {
            "pizzaName": "Salami",
            "additionalToppings": ["🥓", "🌶"]
        },
        {
            "pizzaName": "Margherita",
            "additionalToppings": ["🍌", "🌶", "🍍"]
        }
    ]
}
```

JSON


```
{
    "id": 778,
    "shop": "Luigis Pizza",
    "name": "Edward Olson",
    "phoneNumbers":
        ["(935)503-3765x4154","(935)12345"],
    "address": "Unit 9398 Box 2056\nDPO AP 24022",
    "image": null,
    "pizzas": [
        {
            "pizzaName": "Salami",
            "additionalToppings": ["🥓", "🌶"]
        },
        {
            "pizzaName": "Margherita",
            "additionalToppings": ["🍌", "🌶", "🍍"]
        }
    ]
}
```

JSON

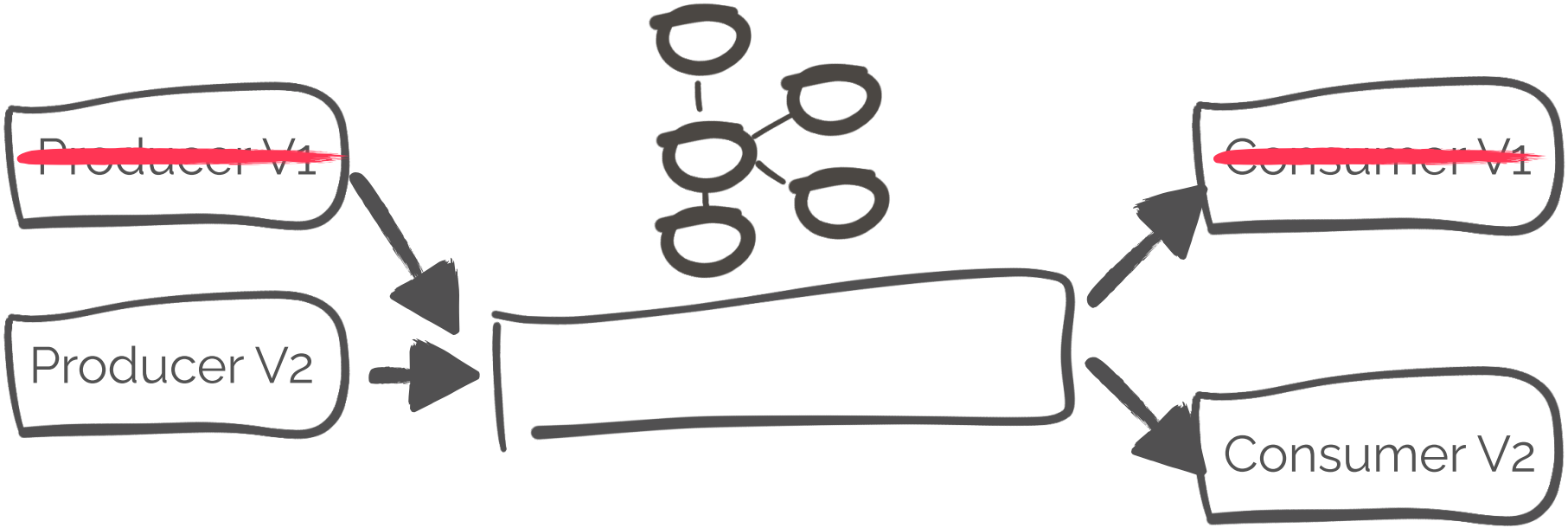
```
{
    "id": 778,
    "shop": "Luigis Pizza",
    "firstname": "Edward",
    "lastname": "Olson",
    "phoneNumbers":
        ["(935)503-3765x4154","(935)12345"],
    "address": "Unit 9398 Box 2056\nDPO AP 24022",
    "image": null,
    "pizzas": [
        {
            "pizzaName": "Salami",
            "additionalToppings": ["🥓", "🌶"]
        },
        {
            "pizzaName": "Margherita",
            "additionalToppings": ["🍌", "🌶", "🍍"]
        }
    ]
}
```


Consumer

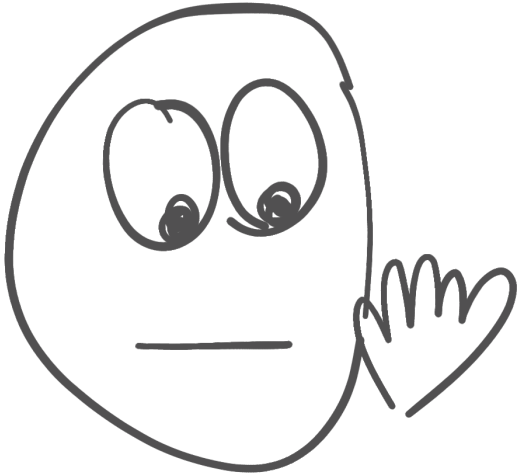
Tight Coupling

Demo 1

Evolution


Producer V1
Producer V2
Consumer V1
Consumer V2

Allowed Change



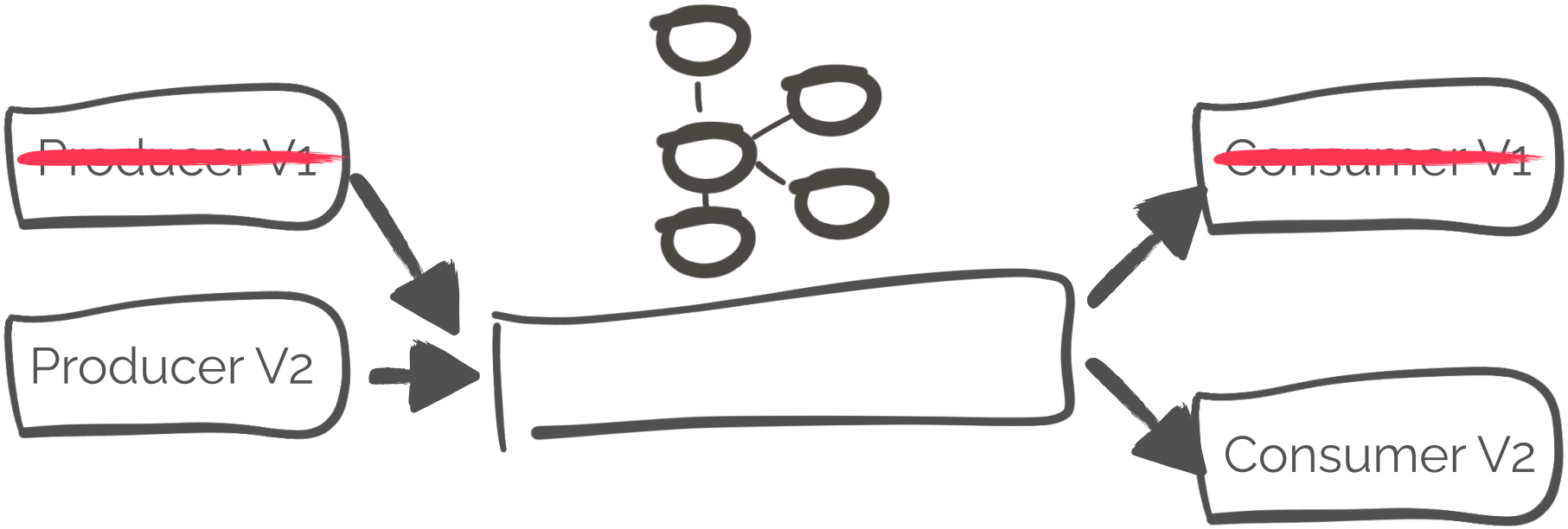
Forbidden Change

Schema Registry

Topic A

Schema Compatibility

Compatibility



COMPATIBILITY = NONE

V_{X-2} V_{X-1} V_X V_{X+1} V_{X+2}

No checks → Like JSON

COMPATIBILITY = BACKWARD

Allowed Changes

- Delete Fields
- Add Optional Fields

Consumers using the V4 schema to decode the message, will be able to also parse messages sent with V3 schema

COMPATIBILITY = BACKWARD_TRANSITIVE

Allowed Changes

- Delete Fields
- Add Optional Fields

Consumers using the V4 schema to decode the message, will be able to also parse messages sent with V3, V2, V1 schema

COMPATIBILITY = FORWARD

V_2 V_3 V_4 V_5 V_6

Allowed Changes

- Add Fields
- Delete Optional Fields

Consumers using V4 schema will be able to also parse messages produced with V5 schemas

COMPATIBILITY = FORWARD_TRANSITIVE

Allowed Changes

- Add Fields
- Delete Optional Fields

Consumers using V4 schema will be able to also parse messages produced with V5, V6, ... schemas

COMPATIBILITY = FULL

Allowed Changes

- Add Optional Fields
- Delete Optional Fields

COMPATIBILITY = FULL_TRANSITIVE

Allowed Changes

- Add Optional Fields
- Delete Optional Fields

Compatibility Summary

BACKWARD

"The consumer i just started, will be able to read all the history"

If you want to be sure to read the past

FORWARD

The consumer i just started, will be able to read all the future

If you don't want to break previous consumers

Schema Registry Summary

Detach Schema from Data

Allow Evolution

Define Compatibility

Schemas... Additional Benefit

Work OOTB with Kafka Connect

Karapace

Schema Registry

REST APIs

Fully Open Source

https://www.karapace.io/

@ftisiot | @aiven_io

Karapace

AVRO JSON

Protobuf

```
from confluent_kafka import Producer
from confluent_kafka.serialization import StringSerializer, SerializationContext, MessageField
from confluent_kafka.schema_registry import SchemaRegistryClient
from confluent_kafka.schema_registry.avro import AvroSerializer
value_str = """
{
"namespace": "example.avro",
"type": "record",
"name": "User",
"fields": [
    {"name": "name", "type": "string"},
    {"name": "age",  "type": ["int", "null"]},
    {"name": "nationality", "type": ["string", "null"], "default":"Italian"}
]
}
"""

def user_to_dict(user, ctx):
    return dict(
        name=user.name,age=user.age,nationality=user.nationality
    )

# Create the Avro serializer
avro_serializer_value = AvroSerializer(
    schema_registry_client, value_str, user_to_dict
)

payload = User(name="John", age=30)
topic ="test-avro"
producer.produce(
    topic=topic,
    value=avro_serializer_value(
        payload,
        SerializationContext(topic, MessageField.VALUE),
    ),
)
```

Producer

```
# Import required libraries
from confluent_kafka import DeserializingConsumer
from confluent_kafka.serialization import SerializationContext, MessageField, StringSerializer
from confluent_kafka.schema_registry import SchemaRegistryClient
from confluent_kafka.schema_registry.avro import AvroDeserializer


avro_deserializer_value = AvroDeserializer(
    schema_registry_client, value_str, dict_to_user
)


# Subscribe to the Kafka topic
consumer.subscribe([env.TOPIC_PREFIX + '-avro'])

# Consume messages from the Kafka topic
while True:
    try:
        msg = consumer.poll(1.0)

        if msg is None:
            continue

        user = avro_deserializer_value(
            msg.value(), SerializationContext(msg.topic(), MessageField.VALUE)
        )

        if user is not None and key is not None:
            print("Key --> {}\n User record --> name: {} age: {}\n".format(key.id, user.name, user.age)
            )

    except KeyboardInterrupt:
            break
    consumer.commit()
```

Consumer

How to Kafka Connect

```
"value.converter": "io.confluent.connect.avro.AvroConverter",
"value.converter.schema.registry.url": "[HOST]:[PORT]",
"value.converter.basic.auth.credentials.source": "USER_INFO",
"value.converter.schema.registry.basic.auth.user.info": "[USERNAME]:[PASSWORD]"
```

Need to enhance throughput?

Keys

Batching

aiven

Enhance Apache Kafka with Schemas

Francesco Tisiot - Senior Developer Advocate

@ftisiot