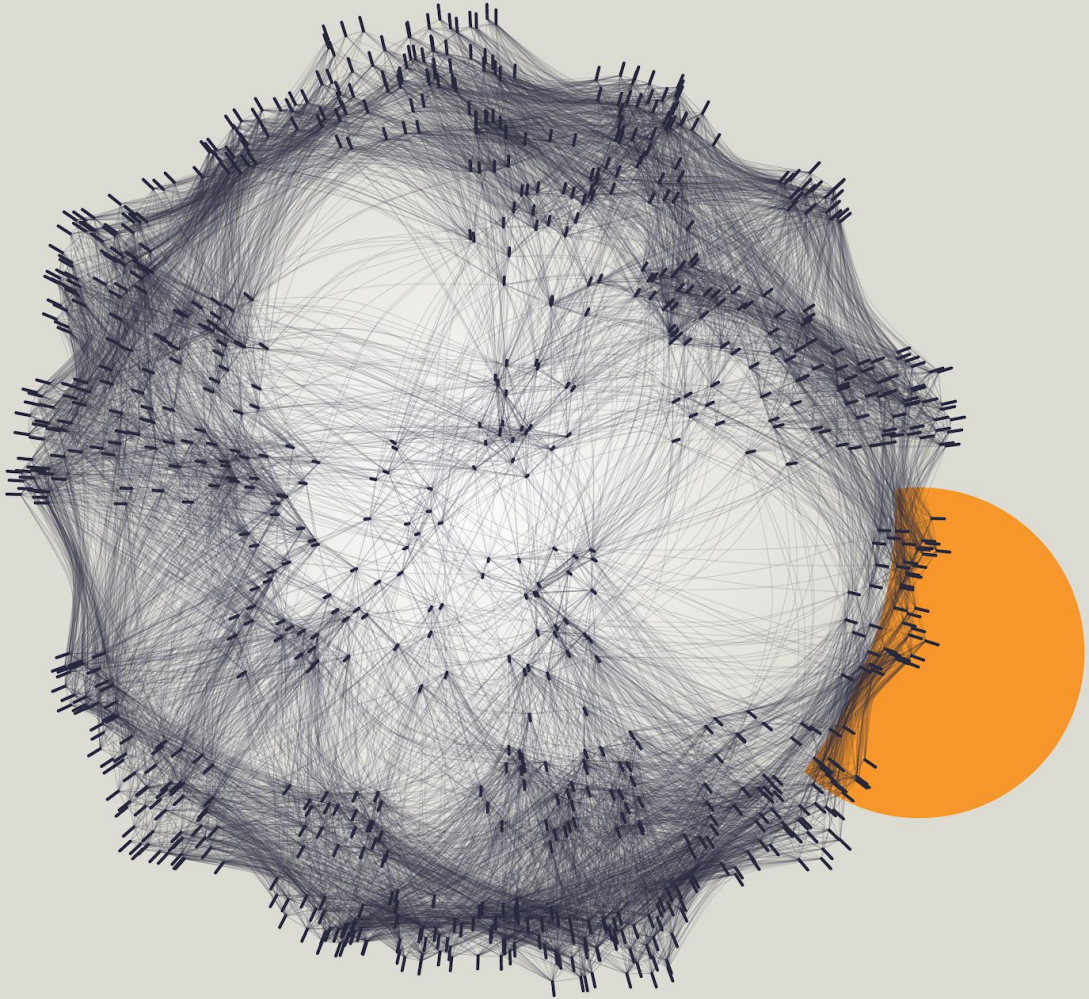


Ever **changing** data model - Schema management for the future

Vidya Venugopal
@venuvid

Software engineer

Founder of thekafkanerd.io

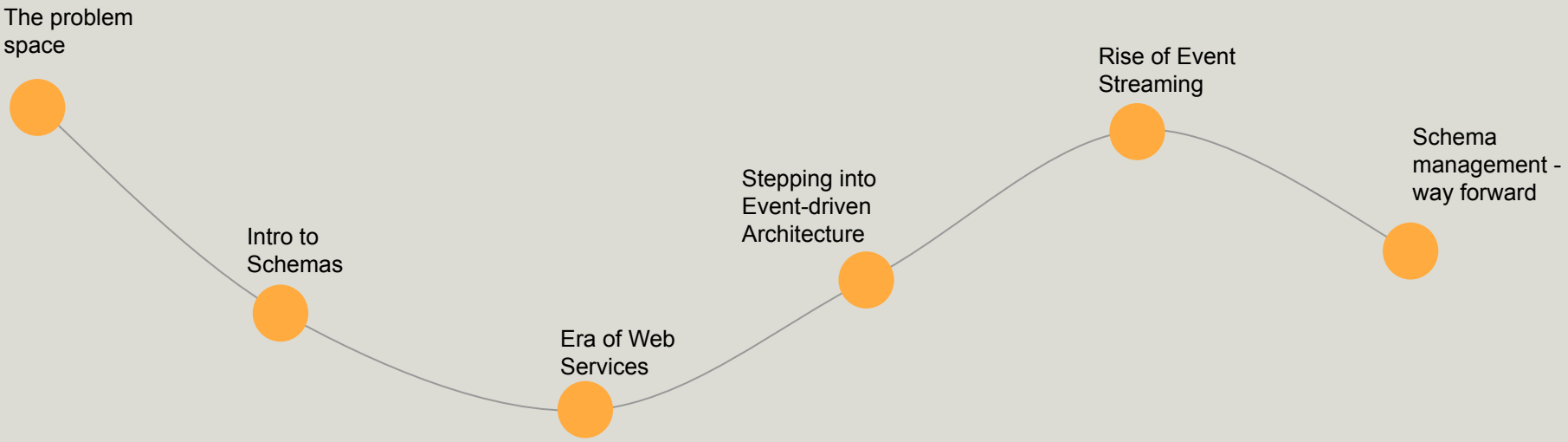
Spent a lot of time on Enterprise Integration

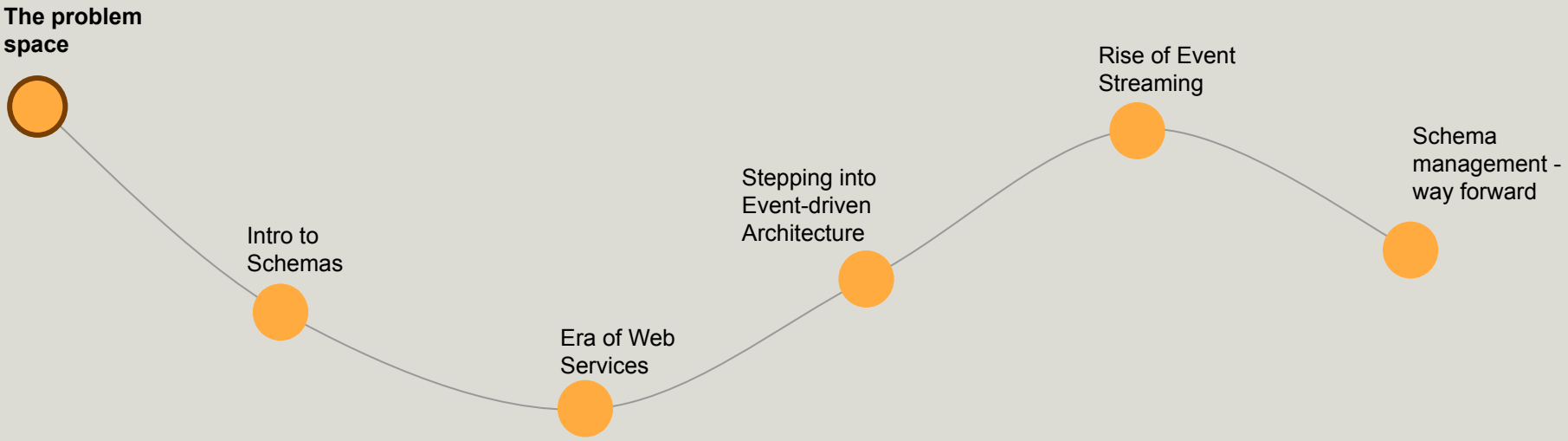
Platforms & Distributed systems

Now, full on with Event Streaming

Senior Dev, Vanguard Australia.

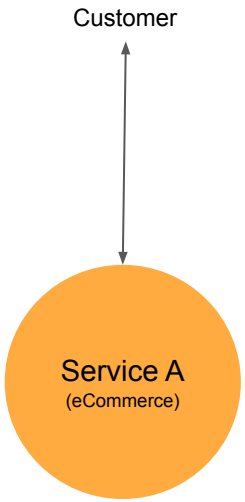
Agenda


The problem space
Intro to Schemas
Era of Web Services
Stepping into Event-driven Architecture
Rise of Event Streaming
Schema management - way forward


The problem space
Intro to Schemas
Era of Web Services
Stepping into Event-driven Architecture
Rise of Event Streaming
Schema management - way forward

The need for Schemas

The need for schemas

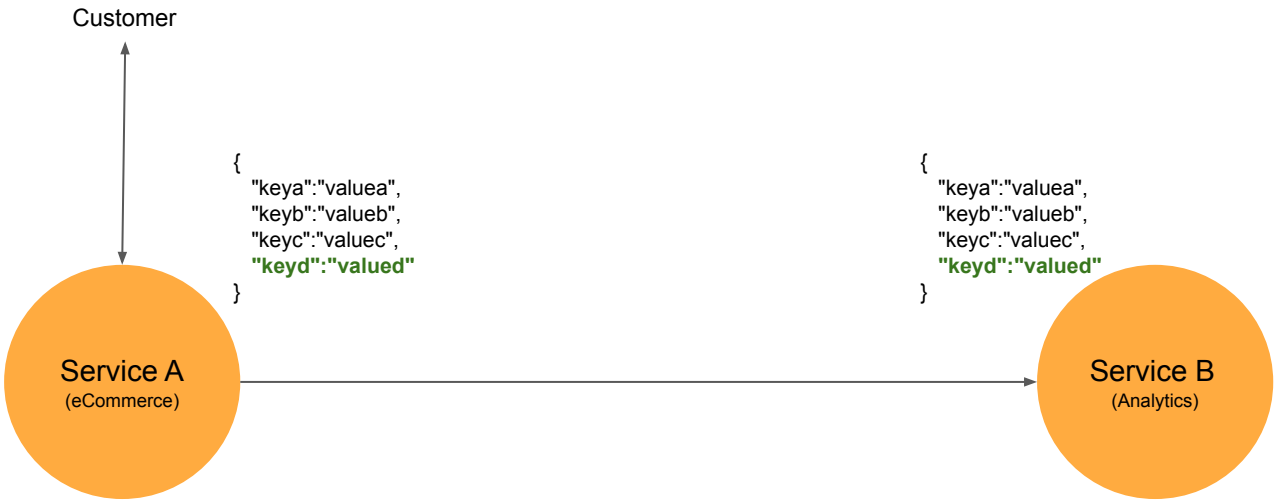


The need for schemas

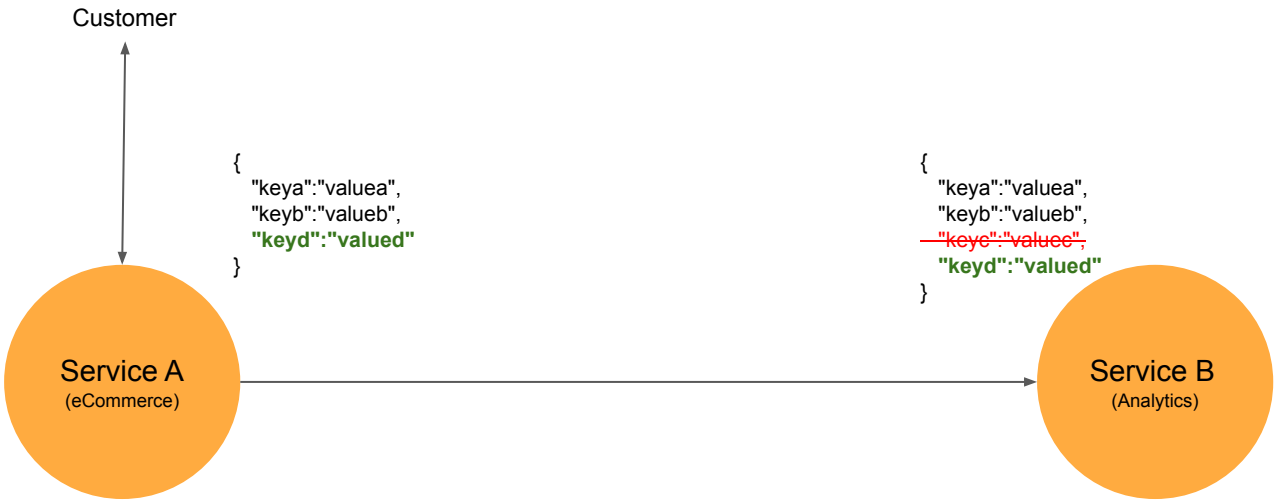
The need for schemas

The need for schemas

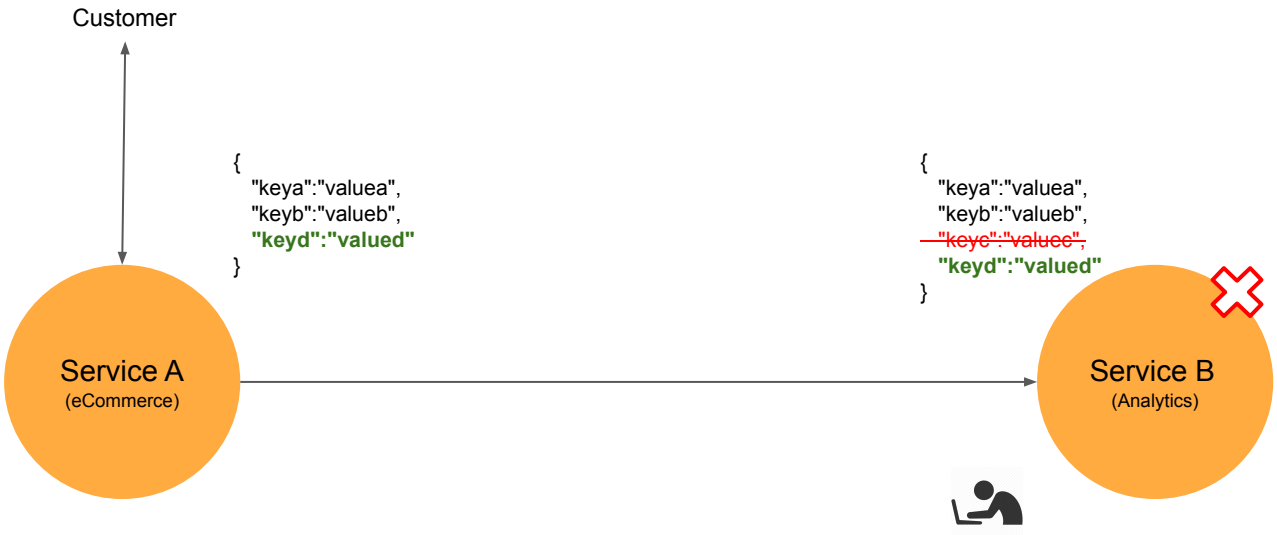
The need for schemas



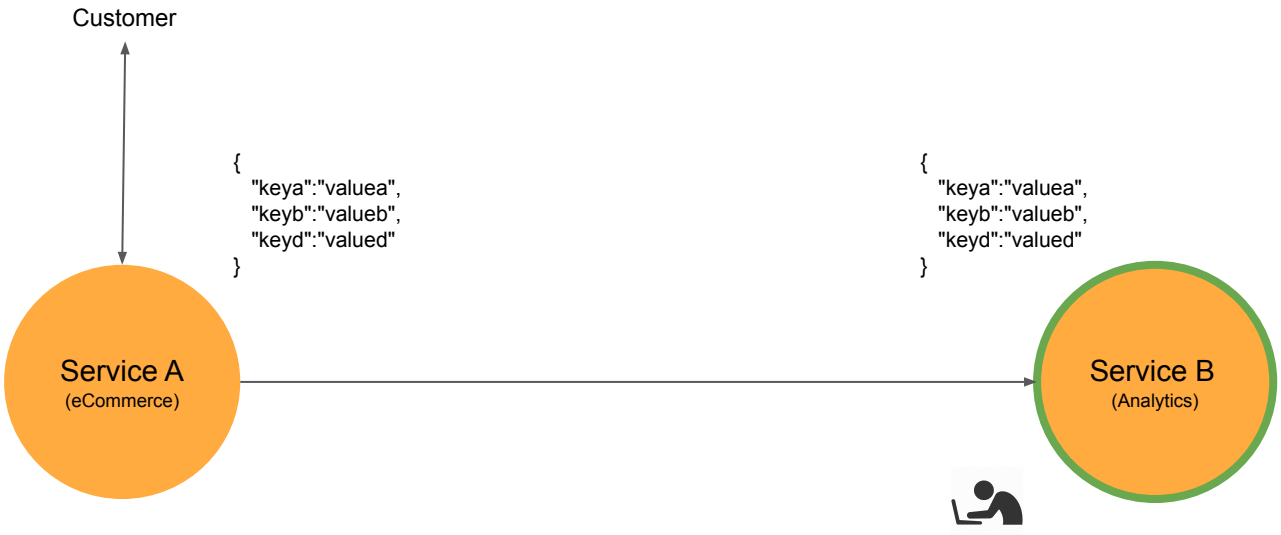
The need for schemas



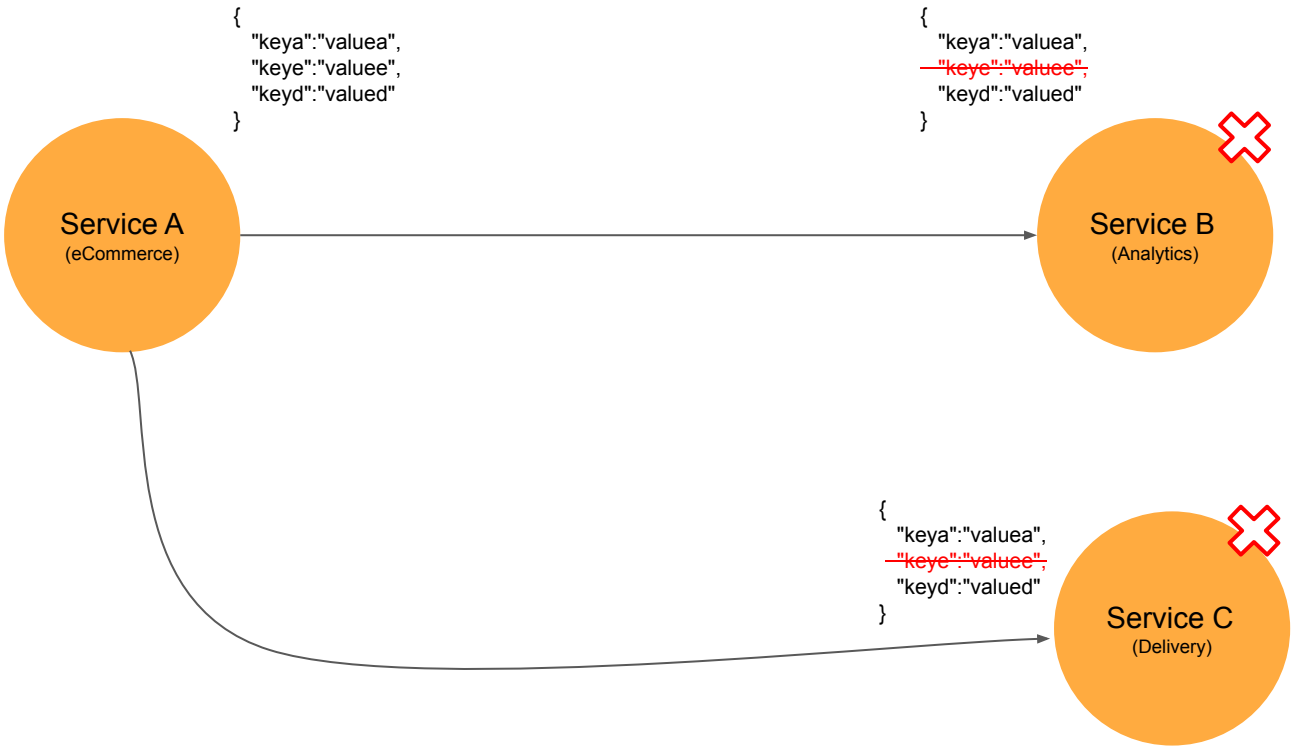
The need for schemas



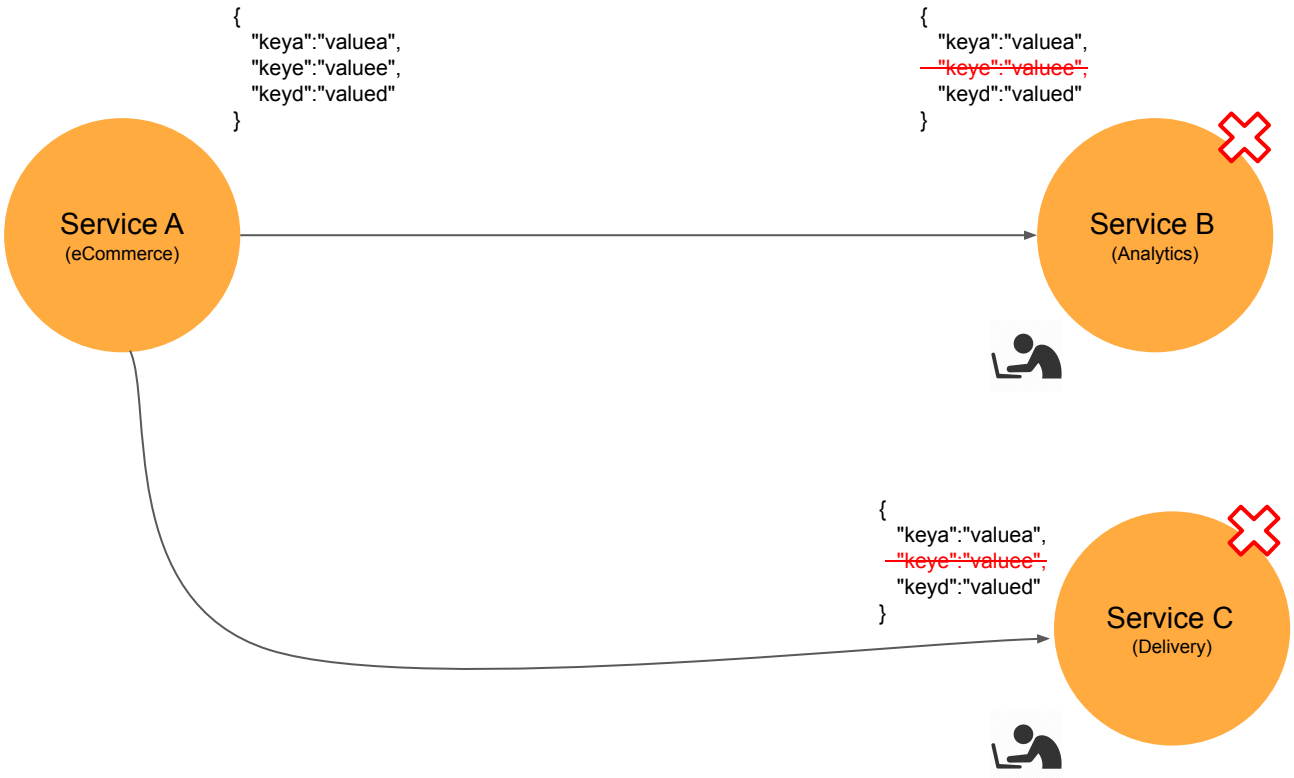
The need for schemas

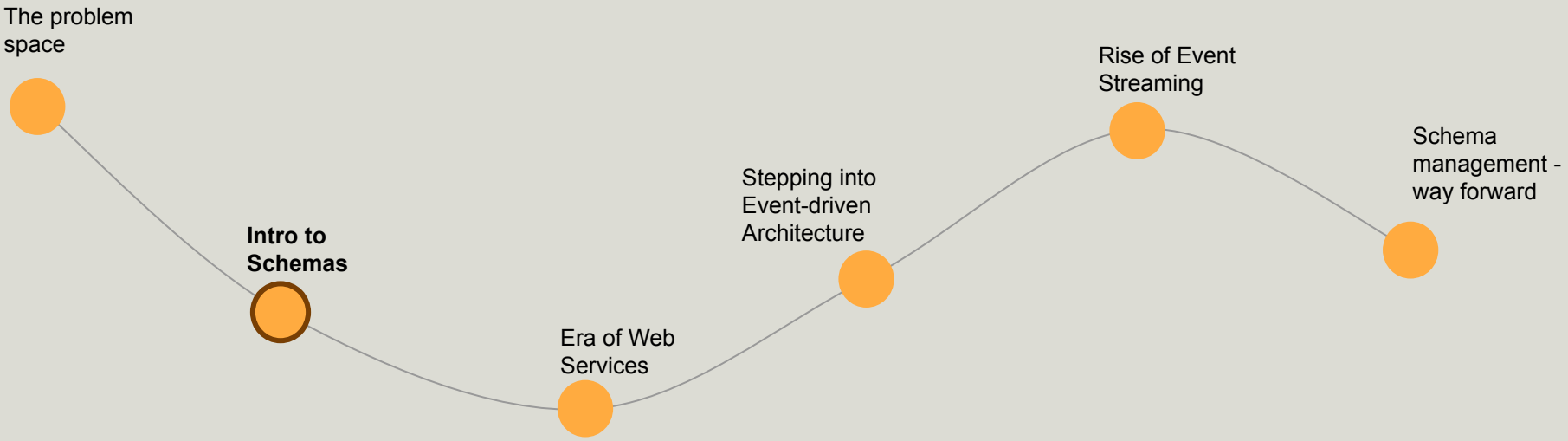


The need for schemas



The need for schemas




The problem space
Intro to Schemas
Era of Web Services
Stepping into Event-driven Architecture
Rise of Event Streaming
Schema management - way forward

Schemas to the rescue

Schemas to the rescue (Examples)

person-schema.xsd

Users > vidya > Desktop > person-schema.xsd

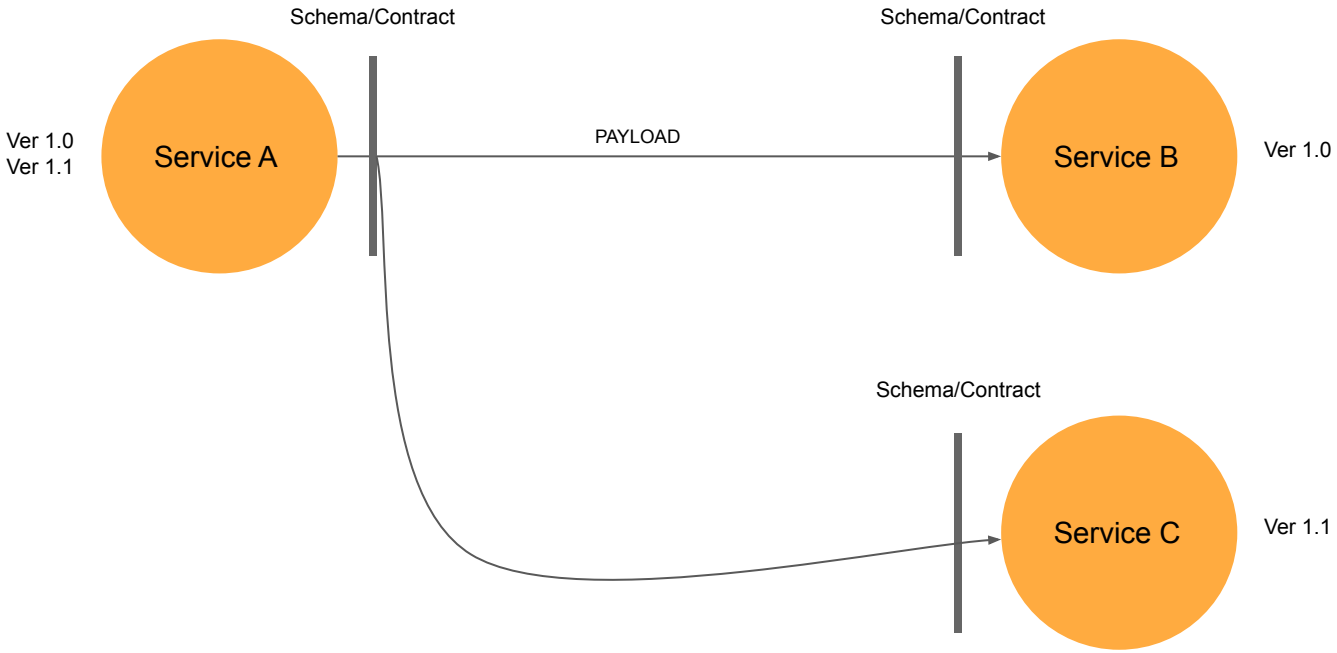
```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="person" type="personType"/>
  <xs:complexType name="personType">
    <xs:sequence>
      <xs:element type="xs:string" name="firstName"/>
      <xs:element type="xs:string" name="lastName"/>
      <xs:element type="xs:integer" name="age"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

person-schema.json

Users > vidya > Desktop > person-schema.json > ...

```json
{
    "$id":"https://example.com/person.schema.json",
    "$schema":"http://json-schema.org/draft-07/schema#",
    "title":"Person",
    "type":"object",
    "properties":{
        "firstName":{
            "type":"string",
            "description":"The person's first name."
        },
        "lastName":{
            "type":"string",
            "description":"The person's last name."
        },
        "age":{
            "description":"Age in years",
            "type":"integer",
            "minimum":0
        }
    }
}
```

Schemas to the rescue



Summary

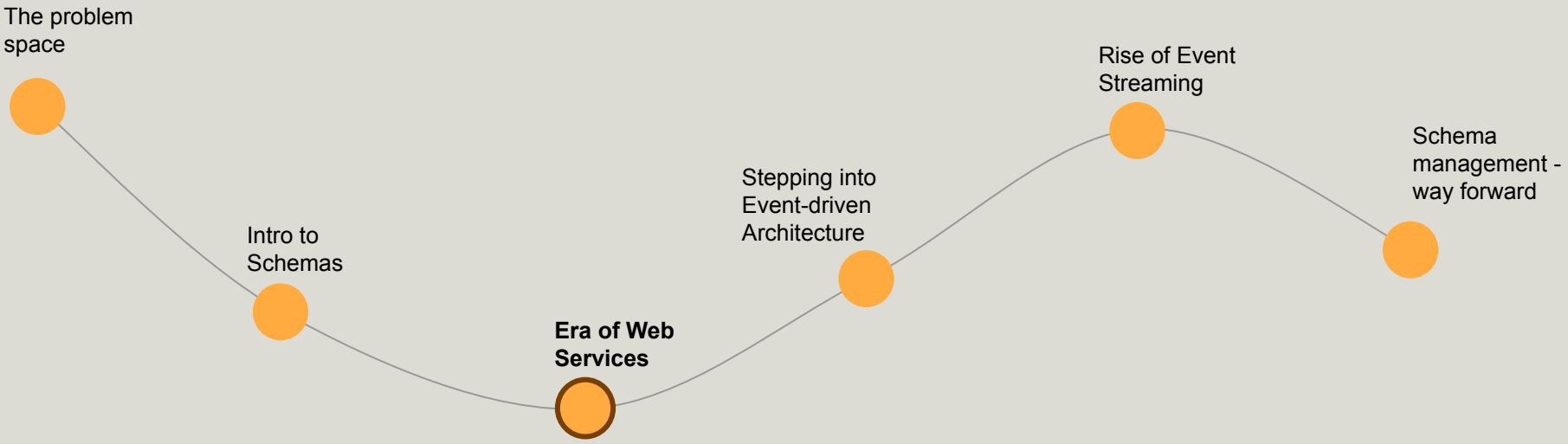
Schemas are the contracts between services

Schemas improve data quality

Schemas allows versioning

It's a way of documenting your interfaces

Enables contract testing


The problem space
Intro to Schemas
Era of Web Services
Stepping into Event-driven Architecture
Rise of Event Streaming
Schema management - way forward

Era of Web Services

Era of Web Services

soapUI Pro 4.6.0 (Licensed to Michael Giller, Smartbear)

File Tools Desktop Try Help

Navigator

- Projects
  - Getting Started
    - WeatherSoap
    - WeatherSoap12
    - TestSuite 1
      - TestCase 1
        - Test Steps (1)
          - GetCityWeatherByZIP - Request 1
        - Load Tests (0)
        - Security Tests (0)

TestCase 1 | GetCityWeatherByZIP - Request 1

http://wsf.cdyne.com/WeatherWS/Weather.asmx

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
   <soapenv:Header/>
   <soapenv:Body>
      <weat:GetCityWeatherByZIP>
         <!--Optional:-->
         <weat:ZIP>02111</weat:ZIP>
      </weat:GetCityWeatherByZIP>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envel
   <soap:Body>
      <GetCityWeatherByZIPResponse xmlns="http://ws.cdyne.com/we
         <GetCityWeatherByZIPResult>
            <Success>true</Success>
            <ResponseText>City Found</ResponseText>
            <State>MA</State>
            <City>Boston</City>
            <WeatherStationCity>Boston</WeatherStationCity>
            <WeatherID>14</WeatherID>
            <Description>Cloudy</Description>
            <Temperature>56</Temperature>
            <RelativeHumidity>86</RelativeHumidity>
            <Wind>N6</Wind>
            <Pressure>30.21S</Pressure>
            <Visibility/>
            <WindChill/>
            <Remarks/>
         </GetCityWeatherByZIPResult>
      </GetCityWeatherByZIPResponse>
   </soap:Body>
</soap:Envelope>
```

Aut | Headers | JMS Property (0) | XSD (string) | Table

Headers (10) | Attachments (0) | SSL Info | WSS (0) | JMS (0) | XSD (?) | Table (1) | XML | Doc | Coverage: ?

Icon to add assertion

SOAP Response - VALID

Click assertions tab

Assertions (1) | Request Log (4)

Response time: 78ms (750 bytes)

9 : 20

soapUI log | http log | jetty log | error log | wsrm log | memory log | script log

TestRequest Properties | Custom Properties

Property	Value
Name	GetCityWeatherByZIP - Req...
Description	
Message Size	335
Encoding	UTF-8
Endpoint	http://wsf.cdyne.com/Weath...
Timeout	
Bind Address	
Follow Redirects	true
Interface	WeatherSoap
Operation	GetCityWeatherByZIP

Properties

Era of Web Services

SMARTBEAR SwaggerHub

SIGN UP LOG IN

open-weather_ma... 2.5.2 Export

Info

Tags

Servers

Search

Current Weather Data

GET /weather

Schemas

SCHEMA 200

SCHEMA Coord

SCHEMA Weather

SCHEMA Main

SCHEMA Wind

SCHEMA Clouds

SCHEMA Rain

SCHEMA Snow

SCHEMA Sys

Aa SAVE

Read Only

```
125          default: "json"
126
127      schemas:
128        200:
129          title: Successful response
130          type: object
131          properties:
132            coord:
133              $ref: '#/components/schemas/Coord'
134            weather:
135              type: array
136              items:
137                $ref: '#/components/schemas/Weather'
138              description: (more info Weather condition codes)
139            base:
140              type: string
141              description: Internal parameter
142              example: cmc stations
143            main:
144              $ref: '#/components/schemas/Main'
145            visibility:
146              type: integer
147              description: Visibility, meter
148              example: 16093
149            wind:
150              $ref: '#/components/schemas/Wind'
151            clouds:
152              $ref: '#/components/schemas/Clouds'
```

OpenWeatherMap API

2.5.2 OAS3

Get current weather, daily forecast for 16 days, and 3-hourly forecast 5 days for your city. Helpful stats, graphics, and this day in history charts are available for your reference. Interactive maps show precipitation, clouds, pressure, wind around your location stations. Data is available in JSON, XML, or HTML format. **Note**: This sample Swagger file covers the `current` endpoint only from the OpenWeatherMap API.

Note: All parameters are optional, but you must select at least one parameter. Calling the API by city ID (using the `id` parameter) will provide the most precise location results.

Terms of service

OpenWeatherMap API - Website

Send email to OpenWeatherMap API

CC Attribution-ShareAlike 4.0 (CC BY-SA 4.0)

API Documentation

Authorize

Era of Web Services

CONSUMER DATA STANDARDS

Search

- Introduction
- Standards
- Security Profile
- Consumer Experience
- Banking APIs
  - Get Accounts
  - Get Bulk Balances
  - Get Balances For Specific A...
  - Get Account Balance
  - Get Account Detail
  - Get Transactions For Account

Get Accounts

`GET /banking/accounts`

Obtain a list of accounts

Endpoint Version

Version	1

Parameters

Name	In	Type	Required	Descriptio
product-category	query	string	optional	Used to filte results on th

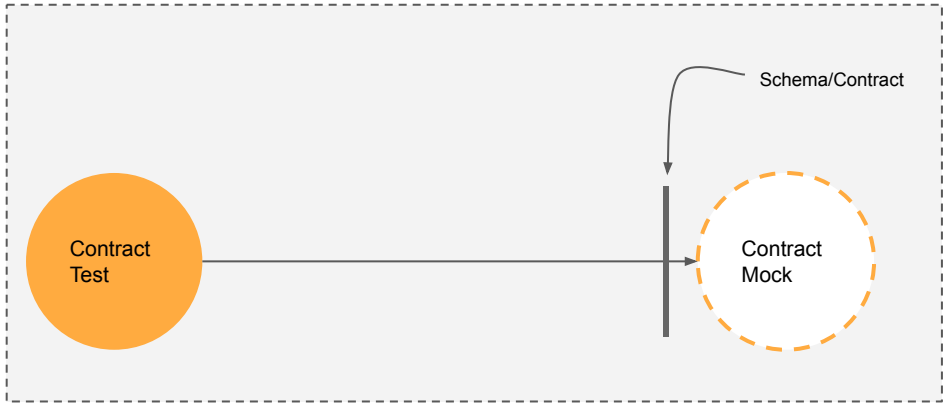
Code samples

```
GET https://data.holder.com.au/cds-au/v1/
Host: data.holder.com.au
Accept: application/json
x-v: string
x-min-v: string
x-fapi-interaction-id: string
x-fapi-auth-date: string
x-fapi-customer-ip-address: string
x-cds-client-headers: string
```

```
var headers = {
  'Accept':'application/json',
```

Contract Testing

- Integration testing, but just with contracts
- Validate the end to end flow without building one.
- Find bugs upfront, don't wait till the actual integration test cycle.

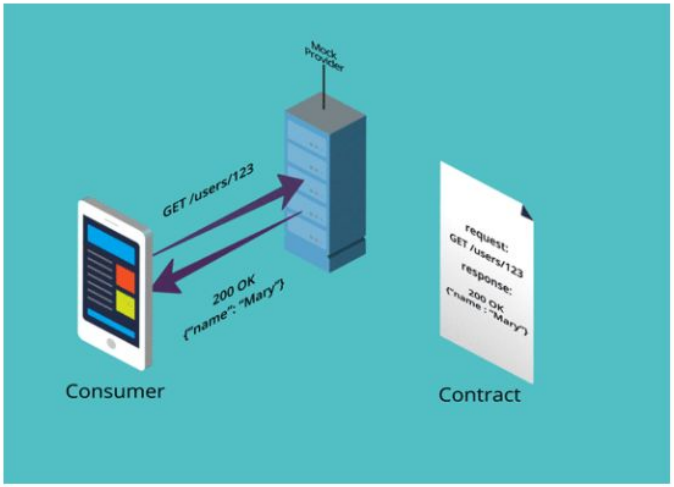


Reference - https://martinfowler.com/bliki/ContractTest.html

Contract Testing - Example

How Pact works

How Pact helps - the consumer side



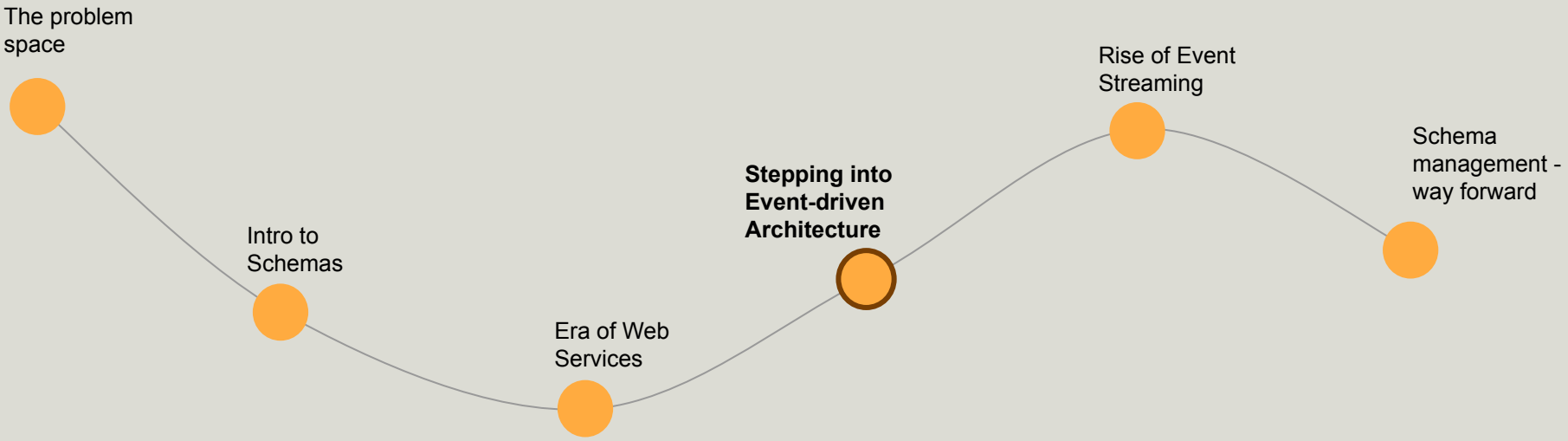
Testing a consumer using a Pact mock provider

Pact solves the problem of keeping the two sets of tests in sync by use of a "contract", also known as a "pact".

During the consumer tests, each request made to a Pact mock provider is recorded into the contract file, along with its expected response.

Replay animation

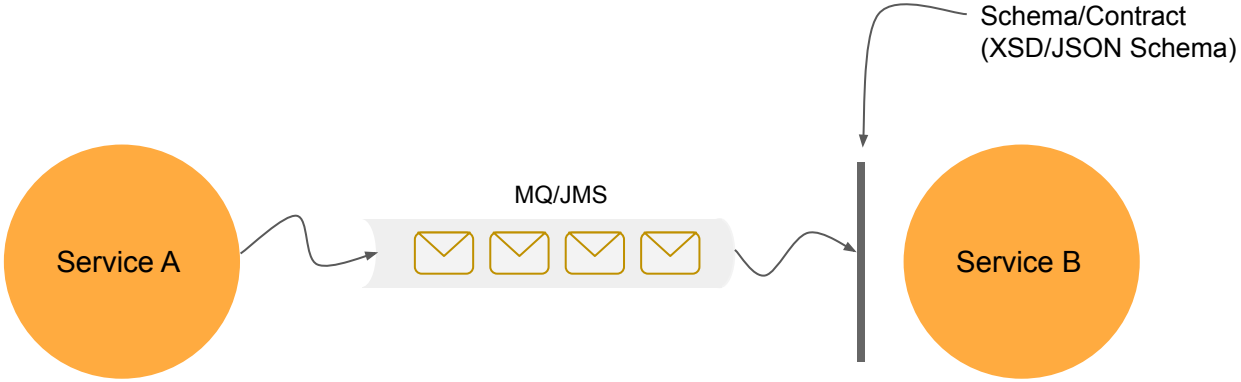
Reference - https://pactflow.io/how-pact-works/#slide-3


The problem space
Intro to Schemas
Era of Web Services
Stepping into Event-driven Architecture
Rise of Event Streaming
Schema management - way forward

Event Driven Architecture

Adoption of Message Queues = Event driven communication + Being real-time + Being asynchronous

Event Driven Architecture



EDA - Schema Validation

ship-order.xml

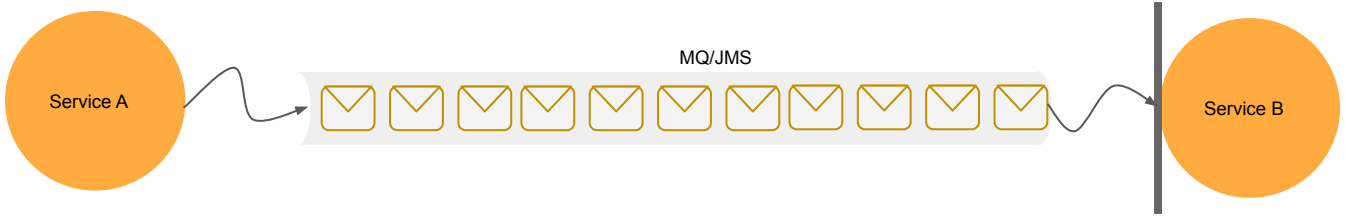
Users > vidya > Desktop > ship-order.xml

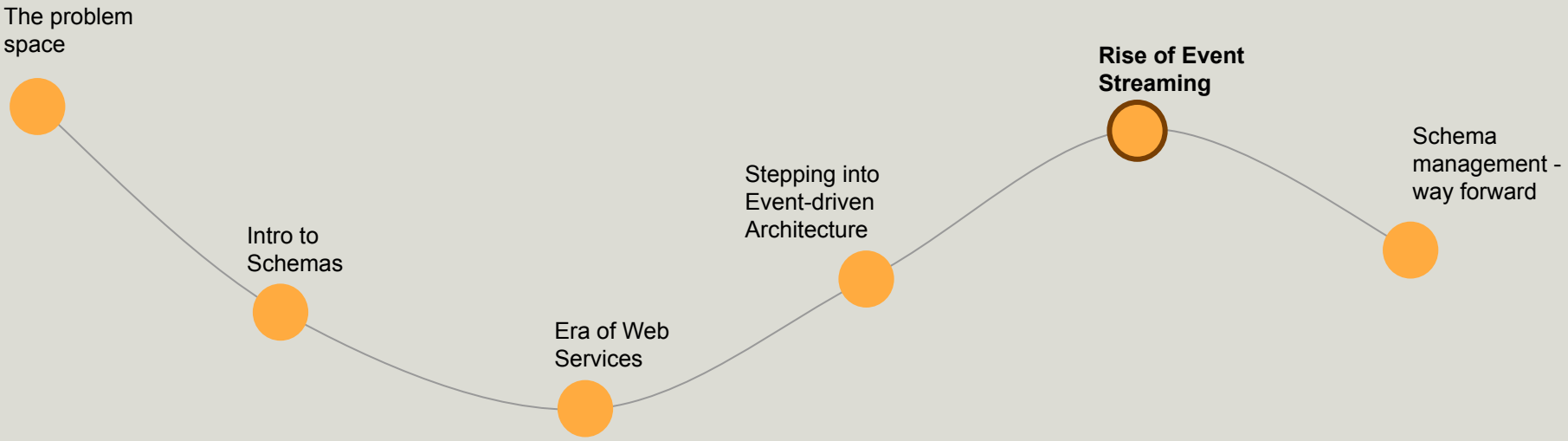
```xml
<?xml version="1.0" encoding="UTF-8"?>

<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
```

ship-order.xsd

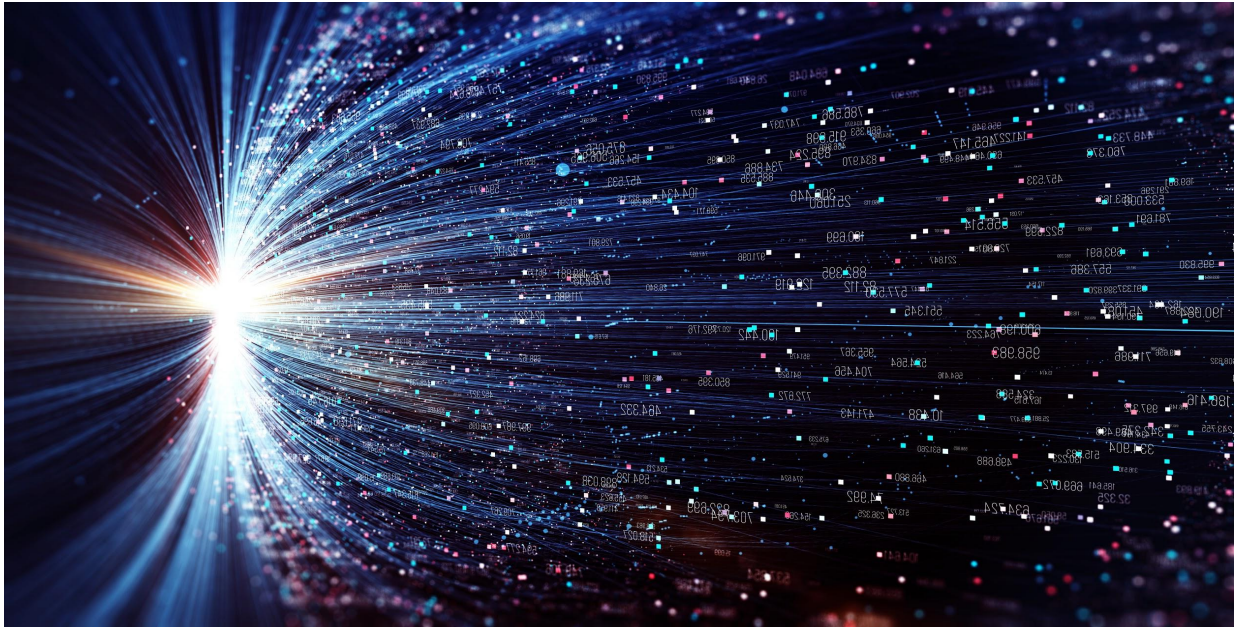
Users > vidya > Desktop > ship-order.xsd

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:simpleType name="stringtype">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="inttype">
        <xs:restriction base="xs:positiveInteger"/>
    </xs:simpleType>
    <xs:simpleType name="dectype">
        <xs:restriction base="xs:decimal"/>
    </xs:simpleType>
    <xs:simpleType name="orderidtype">
        <xs:restriction base="xs:string">
            <xs:pattern value="[0-9]{6}"/>
            </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="shiptotype">
        <xs:sequence>
            <xs:element name="name" type="stringtype"/>
            <xs:element name="address" type="stringtype"/>
            <xs:element name="city" type="stringtype"/>
            <xs:element name="country" type="stringtype"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="itemtype">
        <xs:sequence>
            <xs:element name="title" type="stringtype"/>
            <xs:element name="note" type="stringtype" minOccurs="0"/>
            <xs:element name="quantity" type="inttype"/>
            <xs:element name="price" type="dectype"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="shipordertype">
        <xs:sequence>
            <xs:element name="orderperson" type="stringtype"/>
            <xs:element name="shipto" type="shiptotype"/>
            <xs:element name="item" maxOccurs="unbounded" type="itemtype"/>
        </xs:sequence>
        <xs:attribute name="orderid" type="orderidtype" use="required"/>
    </xs:complexType>
    <xs:element name="shiporder" type="shipordertype"/>
</xs:schema>
```




The problem space
Intro to Schemas
Era of Web Services
Stepping into Event-driven Architecture
Rise of Event Streaming
Schema management - way forward

Event Streaming



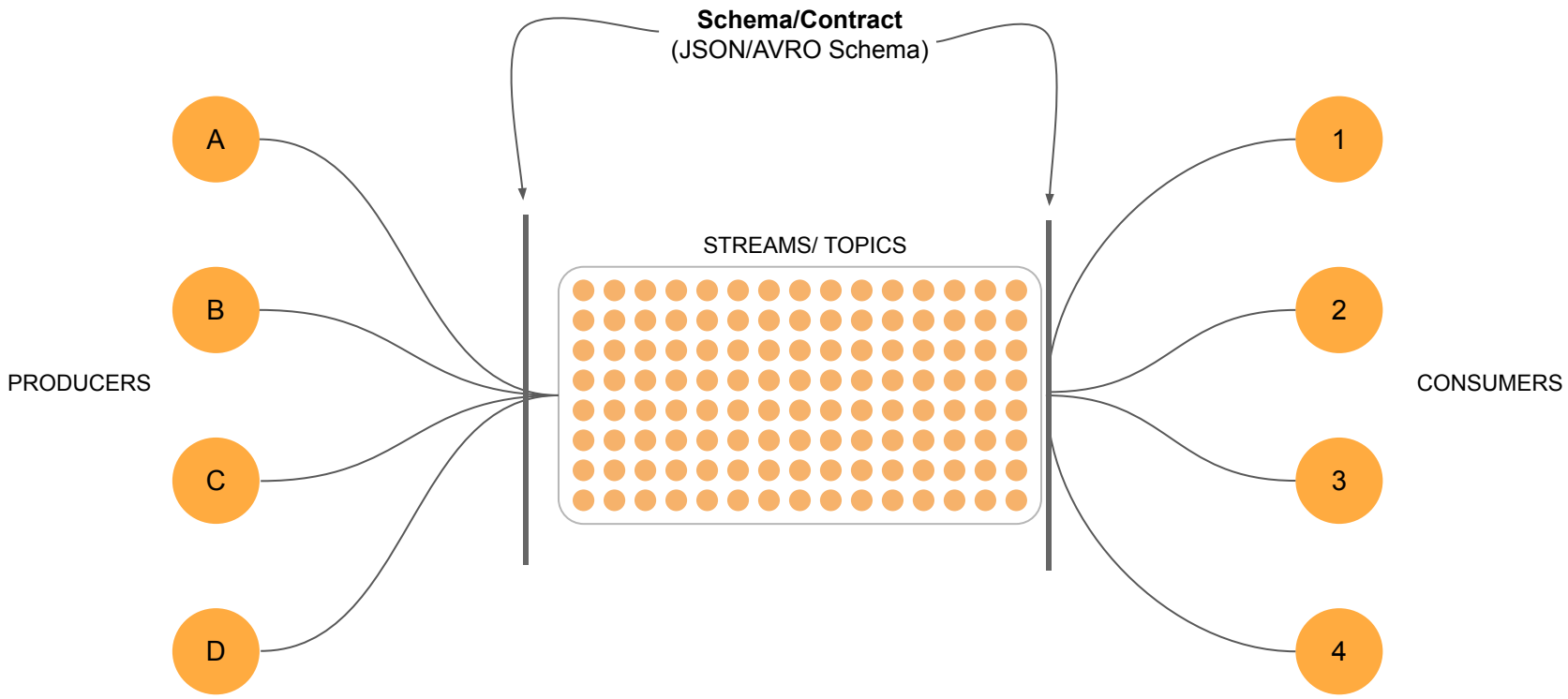
Rise of Event Streams = Capture **All** possible **Events** + at a **Scale** + **Process** them on the fly

Event Streaming - A brief

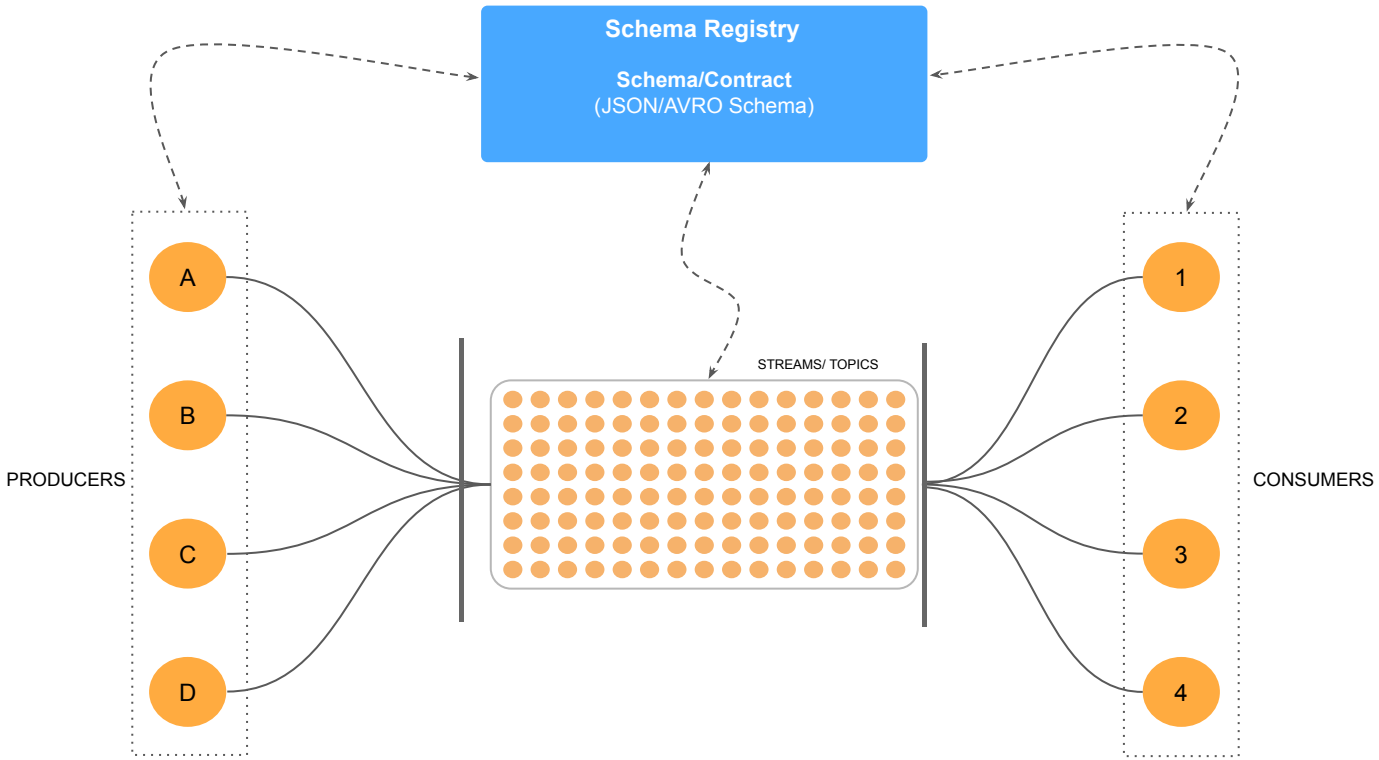
- Record all events at a scale, in and around an Enterprise
- Process/Interrogate/Analyse events on the fly (Event Stream Processing)
- Sophisticated real-time processing abilities: Windowing, re-ordering, etc.
- Good examples: Apache Kafka, AWS Kinesis, Google Pub-Sub

Rise of Schema Registry

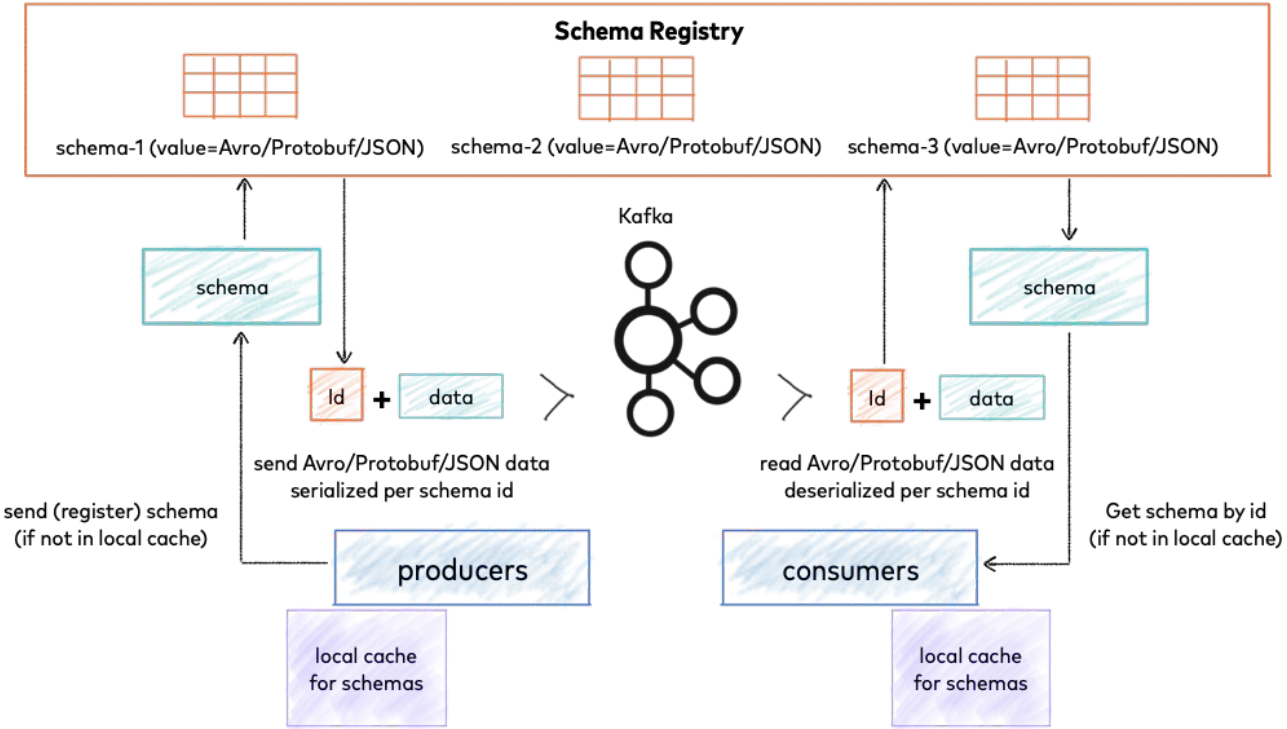
Schema Registry



Schema Registry - How it works



Schema Registry - How it works (Example)



Reference - https://docs.confluent.io/current/schema-registry/index.html

Schema Registry - How it works (Producer Example)

master schema-registry-example / src / main / resources / avro / ParkingAddress.avsc

vidyavenu producer and consumer changes

1 contributor

23 lines (10 sloc) | 331 Bytes

```
{
    "type": "record",
    "namespace": "com.schemaregistryexamples",
    "name": "ParkingAddress",
    "version": "1",
    "fields": [
      { "name": "human_address", "type": "string", "doc": "Complete Address" },
      { "name": "status", "type": "string", "doc": "Parking present or unoccupied" }
    ]
}
```

1. Define Schema

Reference - https://github.com/vidyavenu/schema-registry-example/

Schema Registry - How it works (Producer Example)

2. Register Schema to a topic

ALL TOPICS ›

parking-events

Overview | Messages | Schema | Configuration

Value | Key

Edit schema | Version history | Download

```
1  {
2    "fields": [
3      {
4        "doc": "Complete Address",
5        "name": "human_address",
6        "type": "string"
7      },
8      {
9        "doc": "Parking present or unoccupied",
10       "name": "availability",
11       "type": "string"
12     }
13   ],
14   "name": "ParkingAddress",
15   "namespace": "com.schemaregistryexamples",
16   "type": "record",
17   "version": "1"
18 }
```

Reference - https://github.com/vidyavenu/schema-registry-example/

Schema Registry - How it works (Producer Example)

```java
public class ProducerExample {
    private static final String TOPIC = "parking-events";

    @SuppressWarnings("InfiniteLoopStatement")
    public static void main(final String[] args) {

        final Properties props = new Properties();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
        props.put(ProducerConfig.ACKS_CONFIG, "all");
        props.put(ProducerConfig.RETRIES_CONFIG, 0);
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, KafkaAvroSerializer.class);
        props.put(AbstractKafkaAvroSerDeConfig.SCHEMA_REGISTRY_URL_CONFIG, "http://localhost:8081");

        try (KafkaProducer<String, ParkingAddress> producer = new KafkaProducer<>(props)) {

            for (long i = 0; i < 20; i++) {
                final String parkingID = "parkingid-" + Long.toString(i);

                final ParkingAddress parkingAddress = ParkingAddress.newBuilder().setHumanAddress("Albert st, Melbourne").setStatus("Available").build();
                final ProducerRecord<String, ParkingAddress> record = new ProducerRecord<>(TOPIC, parkingID, parkingAddress);
                producer.send(record);
                Thread.sleep(1000L);
            }
```

3. Produce data

Reference - https://github.com/vidyavenu/schema-registry-example/

Schema Registry - How it works (Producer Example)

```
17:17:52.705 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.clients.NetworkClient - [Producer clientId=producer-1] Completed connec
17:17:52.705 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.clients.NetworkClient - [Producer clientId=producer-1] Initiating API v
17:17:52.713 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.clients.NetworkClient - [Producer clientId=producer-1] Recorded API ver
Successfully sent 20 messages to  parking-events
17:18:12.773 [main] INFO org.apache.kafka.clients.producer.KafkaProducer - [Producer clientId=producer-1] Closing the Kafka producer with timeoutMillis
17:18:12.773 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.clients.producer.internals.Sender - [Producer clientId=producer-1] Begi
17:18:12.784 [kafka-producer-network-thread | producer-1] DEBUG org.apache.kafka.clients.producer.internals.Sender - [Producer clientId=producer-1] Shut
17:18:12.785 [main] DEBUG org.apache.kafka.clients.producer.KafkaProducer - [Producer clientId=producer-1] Kafka producer has been closed
```

4. Messages sent !

Reference - https://github.com/vidyavenu/schema-registry-example/

Schema Registry - How it works (Producer Example)

Update schema - Introduced parking_id

master

Browse files

vidyavenu committed 22 seconds ago Verified

1 parent fc6b2ea commit 3f89fd5bfc0c2f819125a3e236f522699373f41d

Showing 1 changed file with 2 additions and 1 deletion.

Unified Split

3 src/main/resources/avro/ParkingAddress.avsc

```diff
@@ -5,7 +5,8 @@
      "version": "1",
      "fields": [
        { "name": "human_address", "type": "string", "doc": "Complete Address" },
-       { "name": "status", "type": "string", "doc": "Parking present or unoccupied" }
+       { "name": "status", "type": "string", "doc": "Parking present or unoccupied" },
+       { "name": "parking_id", "type": "string", "doc": "Parking ID" }
      ]
  }
```

6. Producer modifies the payload

Reference - https://github.com/vidyavenu/schema-registry-example/

Schema Registry - How it works (Producer Example)

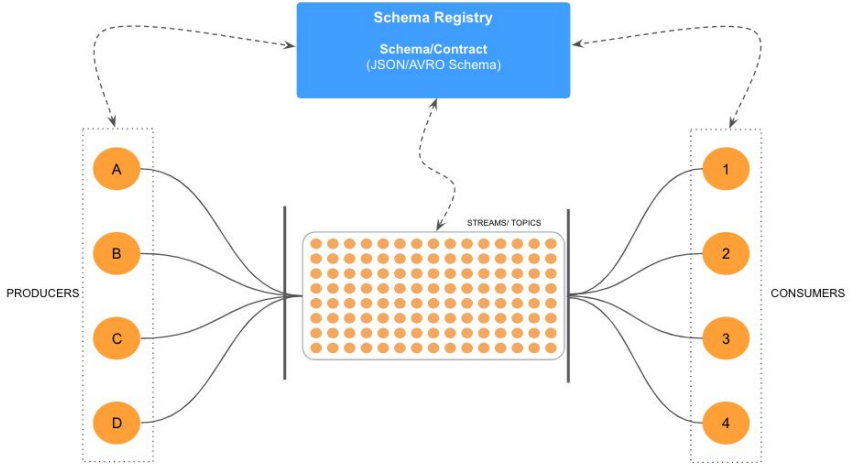
7. Validation fails :)

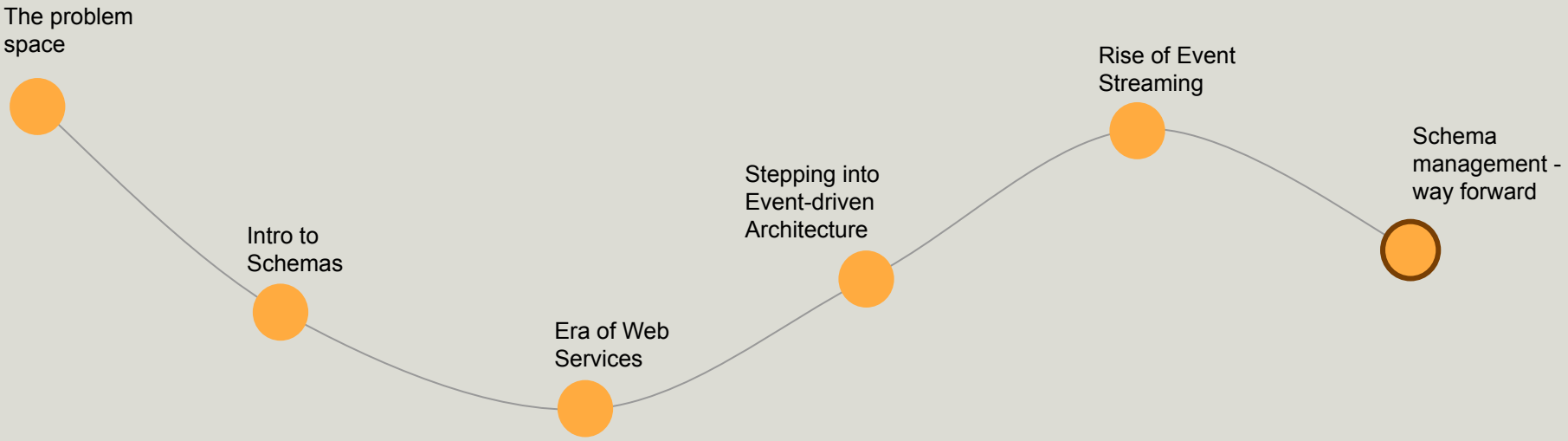
```
Run:  schema-registry-example [ProducerExam...
schema-registry-example [ProducerExample.main()]: successful at 10/8/20, 5:22 pm with 1 error        2 s 167 ms
:ProducerExample.main()  1 error                                                                    1 s 241 ms
io.confluent.kafka.schemaregistry.client.rest.exceptions.RestClientException: Schema being registered is incompatible with an earlier schema; error code: 409
```

Reference - https://github.com/vidyavenu/schema-registry-example/

Schema Registry - Summary

- Centralised schema registry between Producers, Consumers & Event Streams
- Simplified serialization and deserialization
- Allows versioning & compatibility checks (forward/backward compatibility)




The problem space
Intro to Schemas
Era of Web Services
Stepping into Event-driven Architecture
Rise of Event Streaming
Schema management - way forward