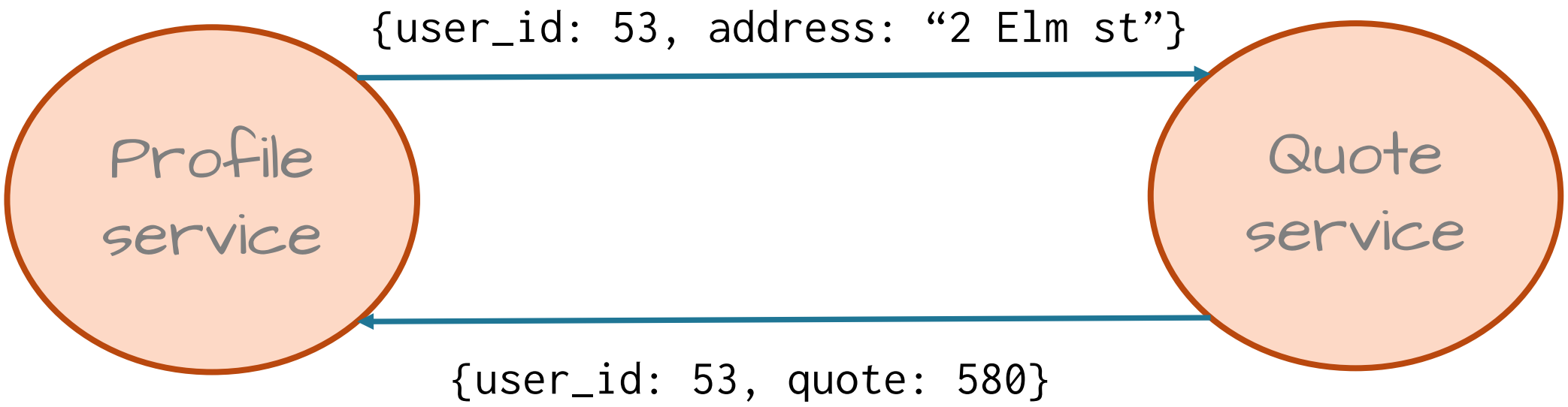
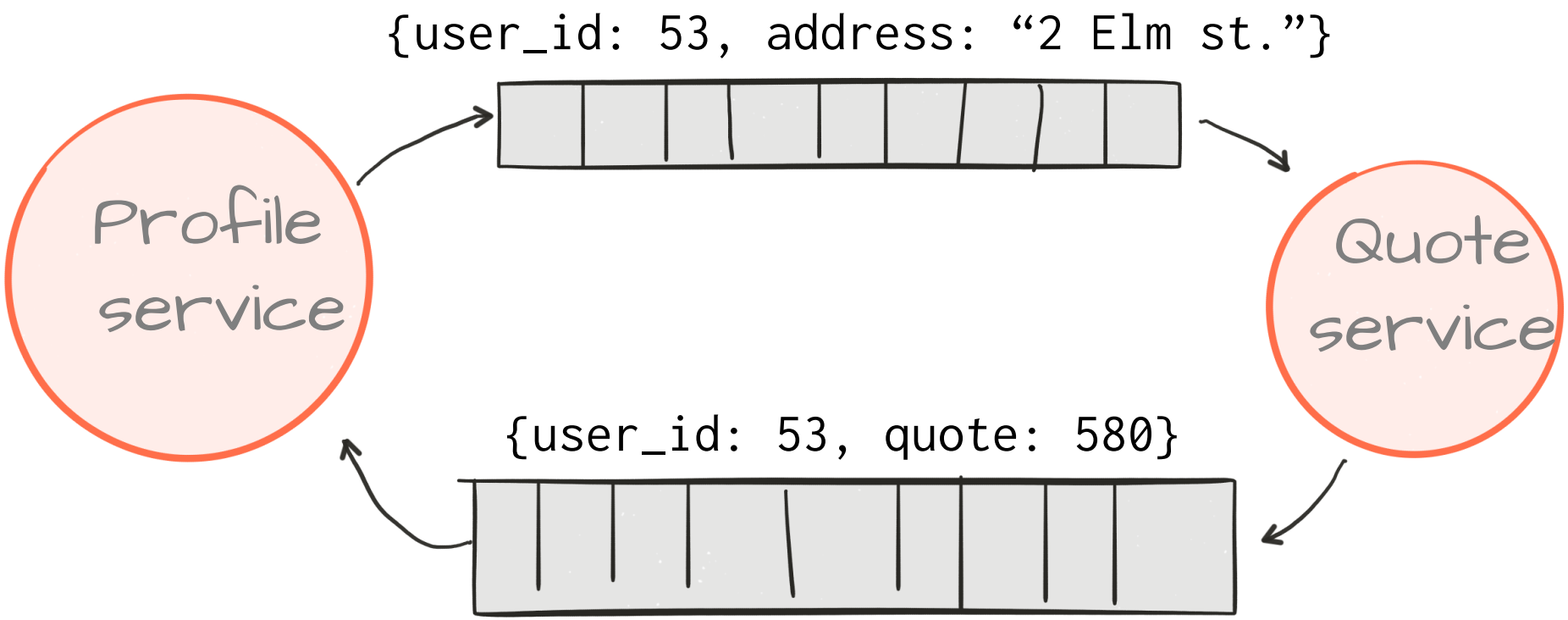
Streaming Microservices: Contracts & Compatibility

Gwen Shapira
Confluent Inc.

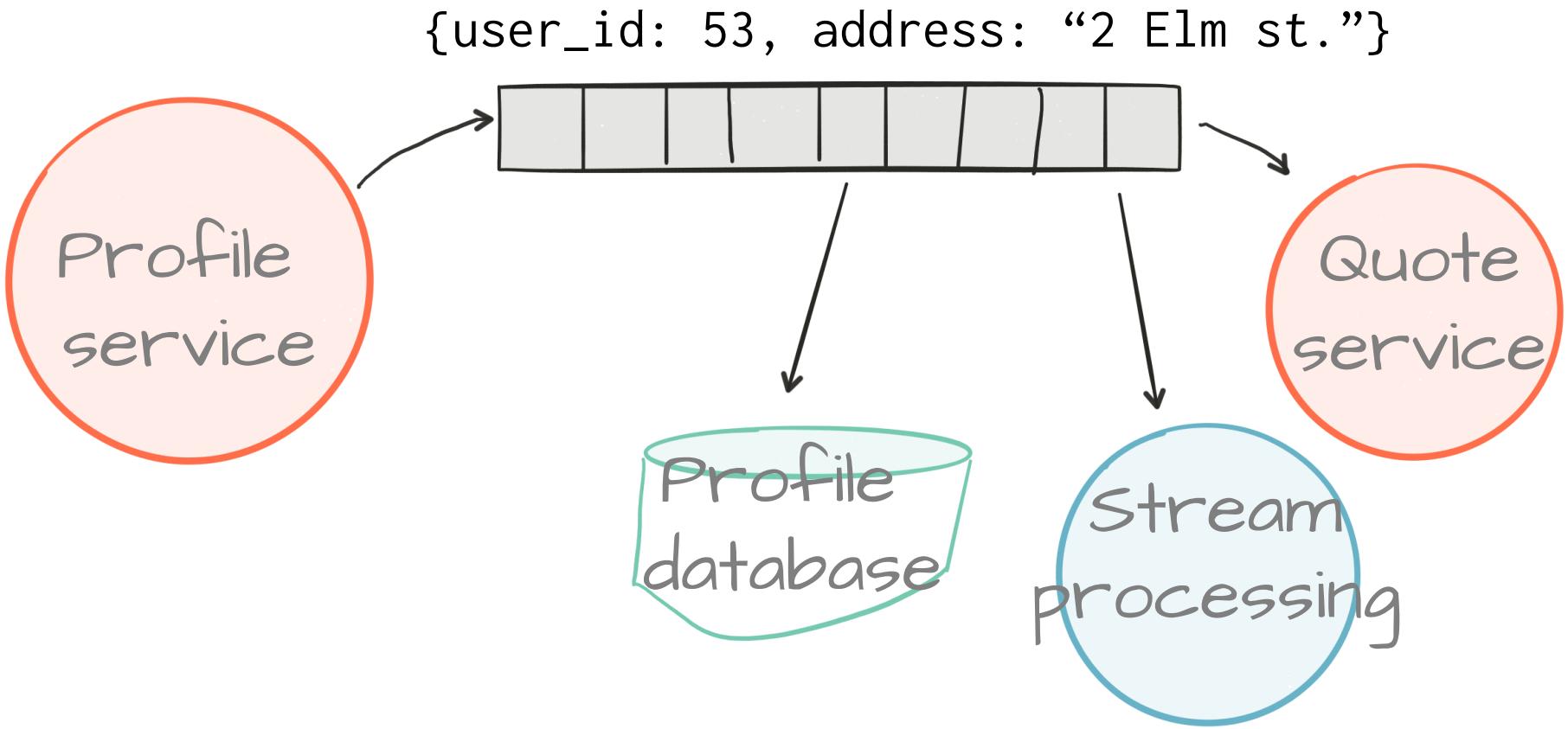
APIs are contracts between services



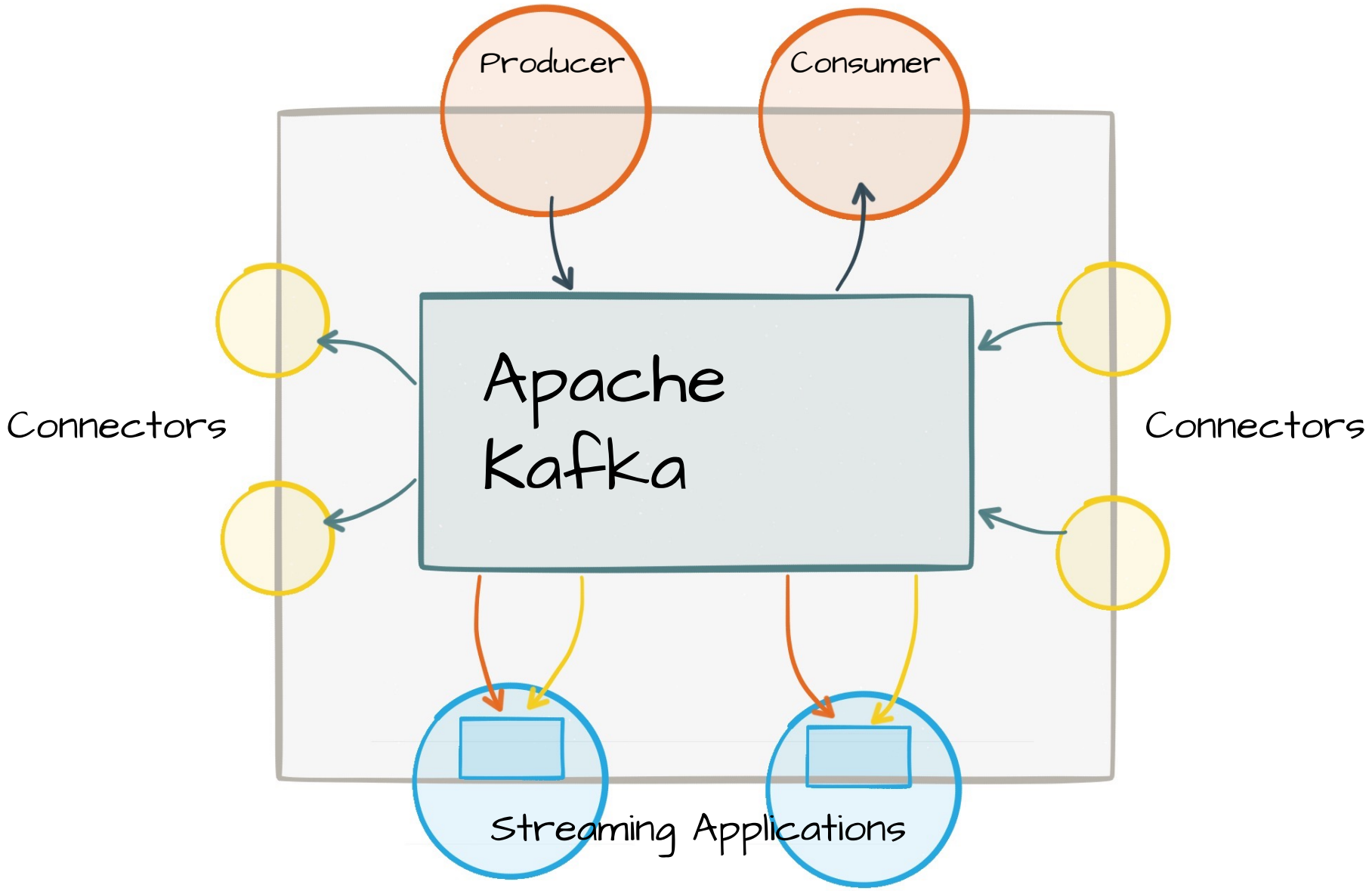
But not all services Talk to each other directly



And naturally...

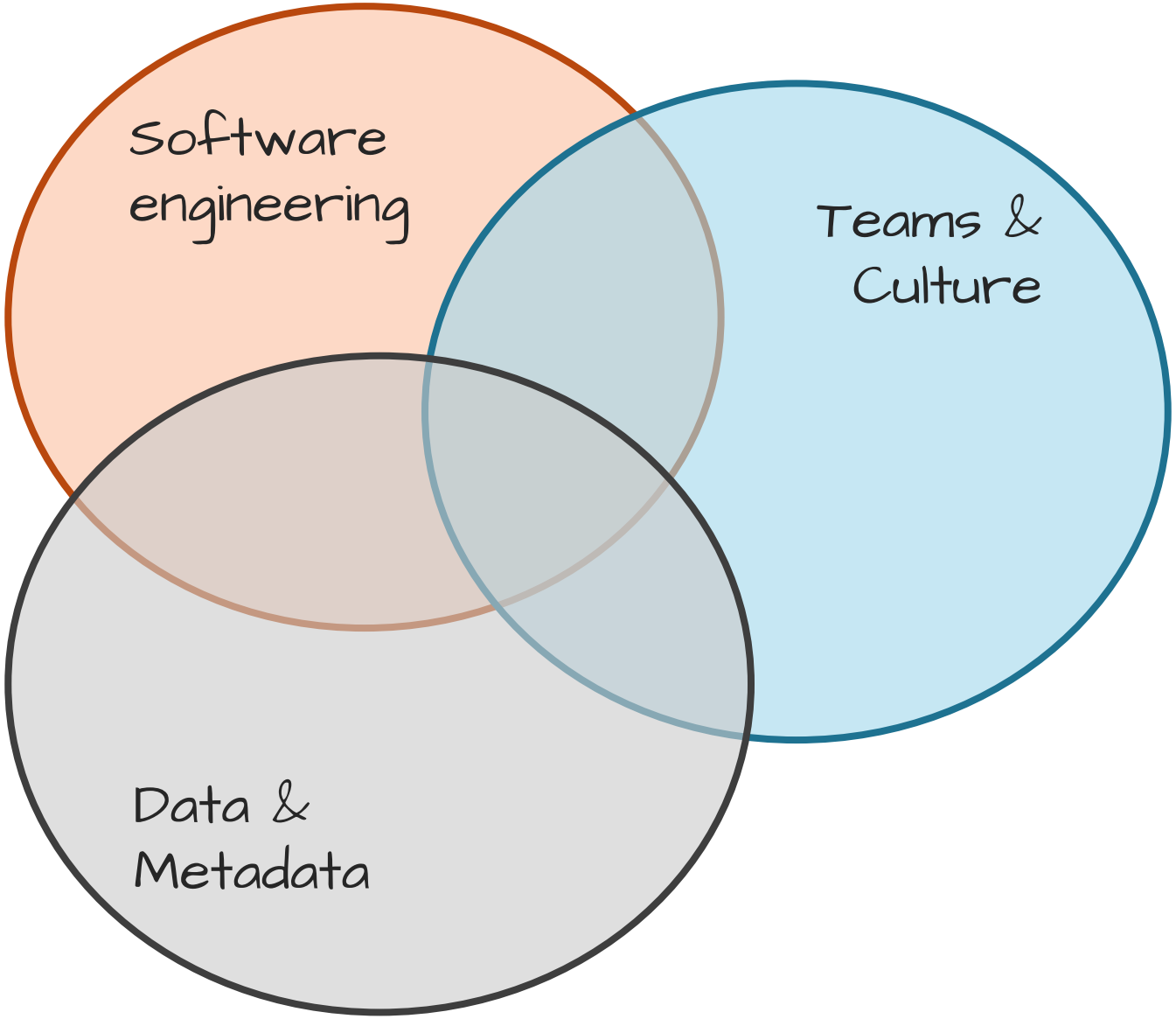


… and then you have a streaming platform



Schema are APIs.

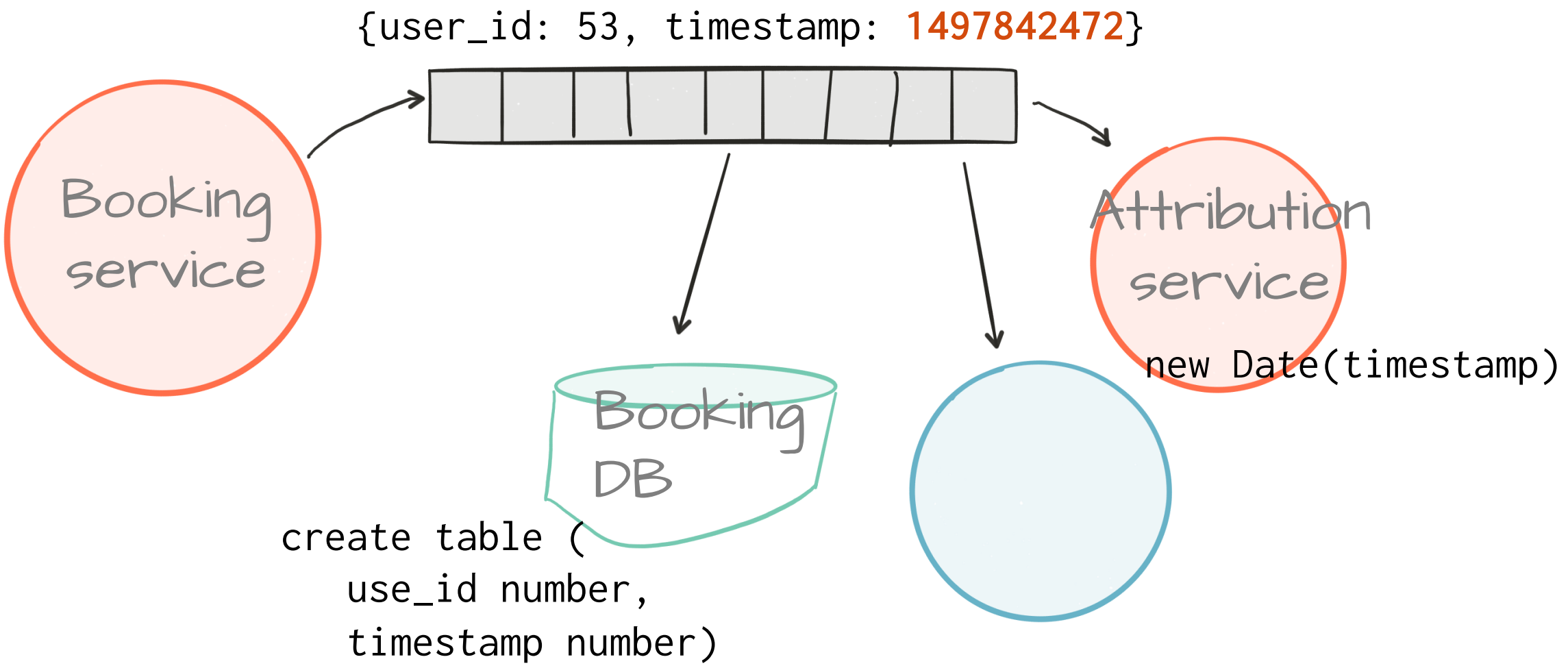
It isn't just about the services

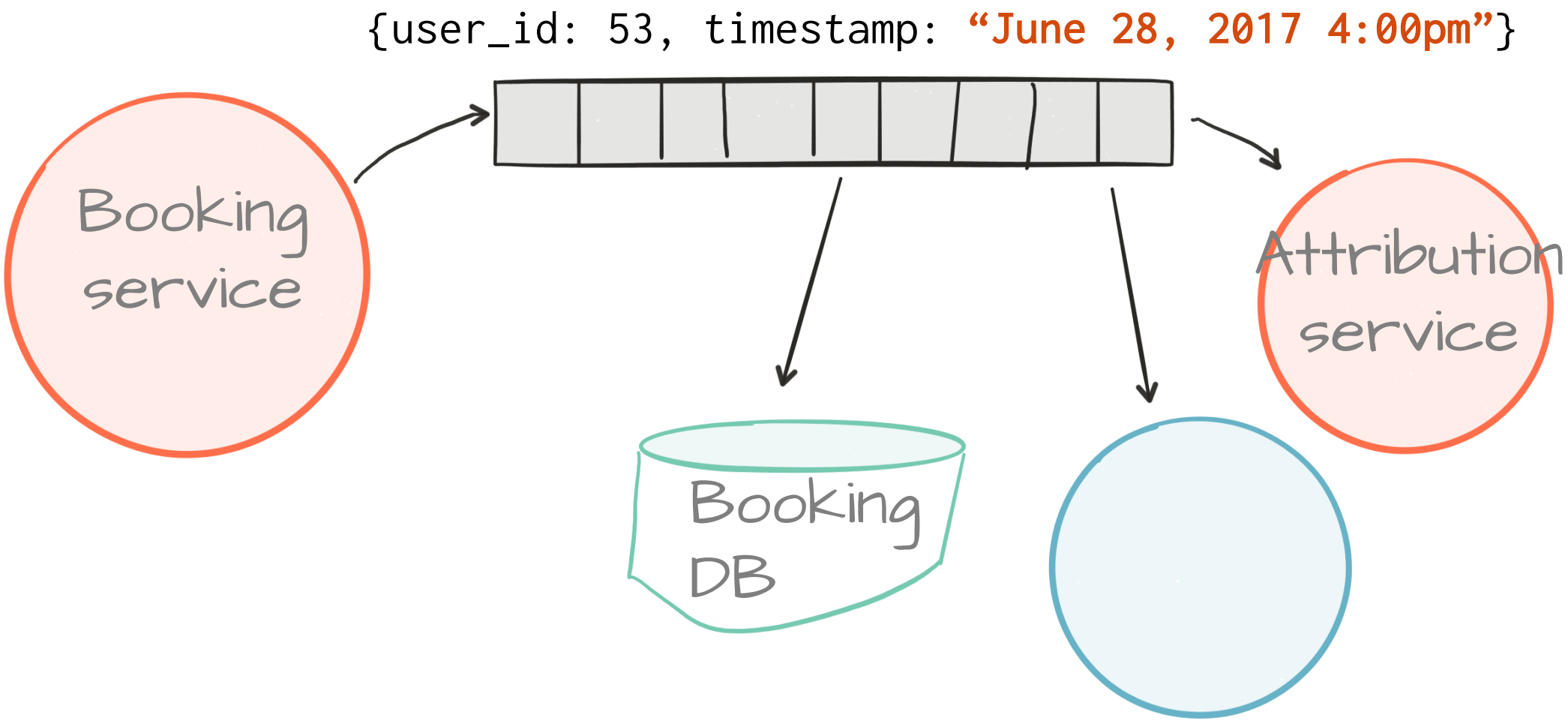


Lack of Schema can tightly couple teams and services

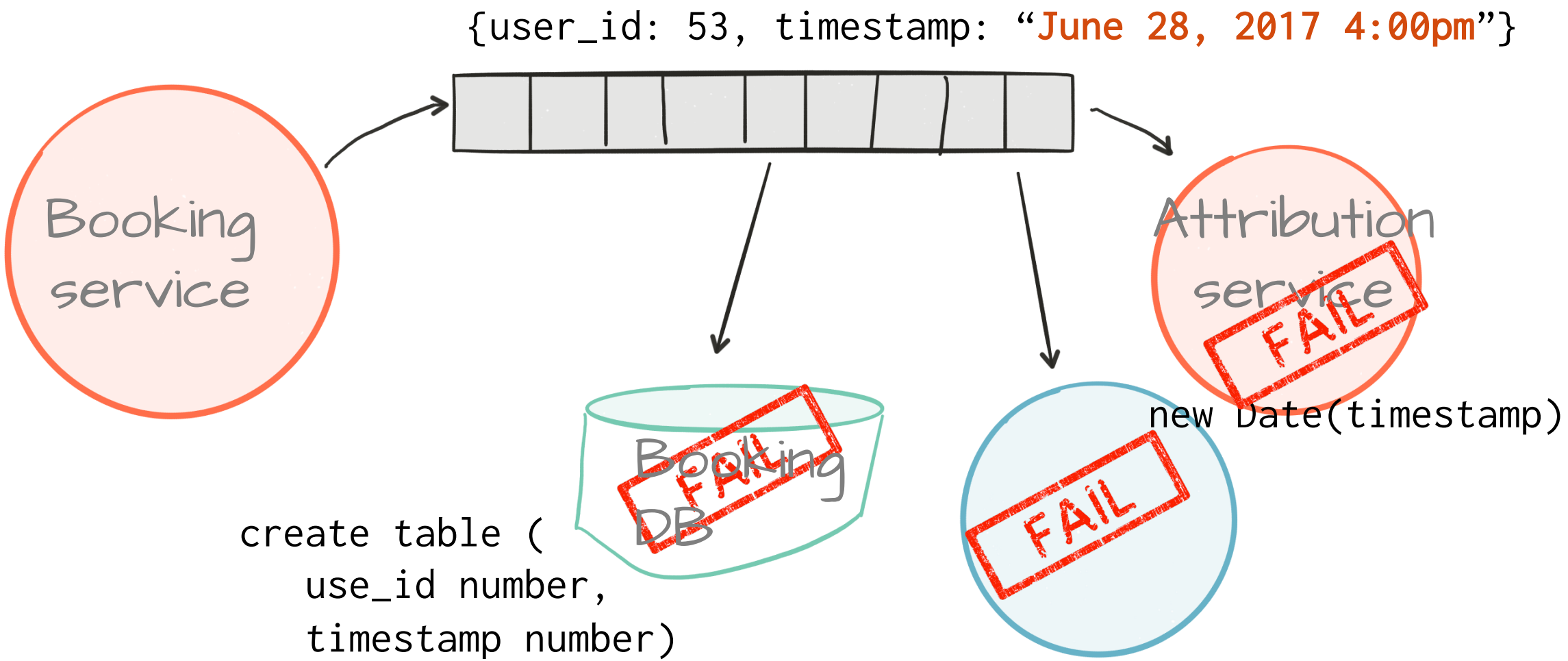
```
2001  2001  Citrus Heights-Sunrise Blvd
   Citrus_Hghts   60670001 3400293  34 SAC
   Sacramento  SV Sacramento Valley SAC
   Sacramento County APCD  SMA8 Sacramento
Metropolitan Area CA 6920  Sacramento  28 6920
   13588 7400 Sunrise Blvd 95610 38 41 56
   38.6988889  121   16 15.98999977   -121.271111
   10 4284781  650345   52
```

Schemas are about how teams work together



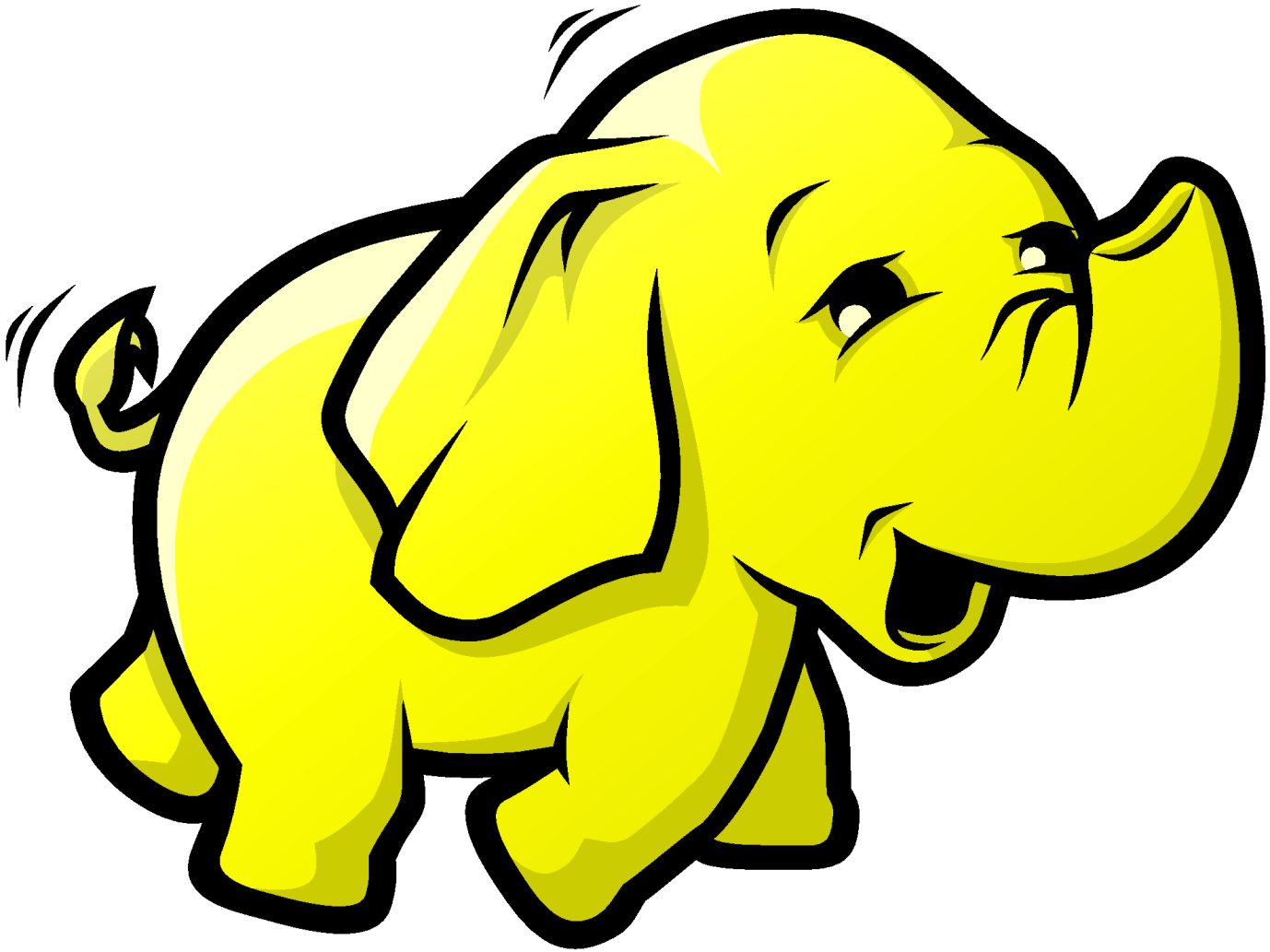

{user_id: 53, timestamp: “June 28, 2017 4:00pm”}
Booking service
Attribution service
Booking DB

Moving fast and breaking things



Back in my day… It was never a problem.

And then it was.

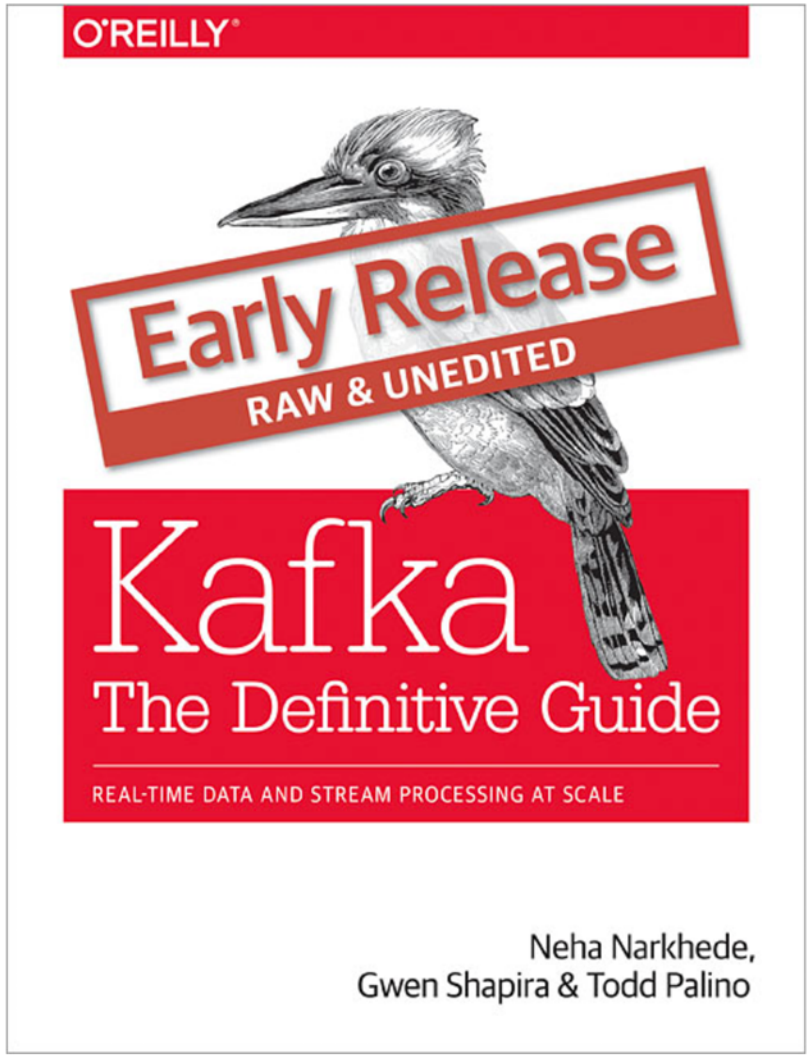


Moving data around since 1997

Missing my Schema since 2012.

Apache Kafka PMC

Tweeting a lot @gwenshap



Existing solutions

Existing solutions

"It is a communication problem"

"We need to improve our process"

"We need to document everything and get stakeholder approval"

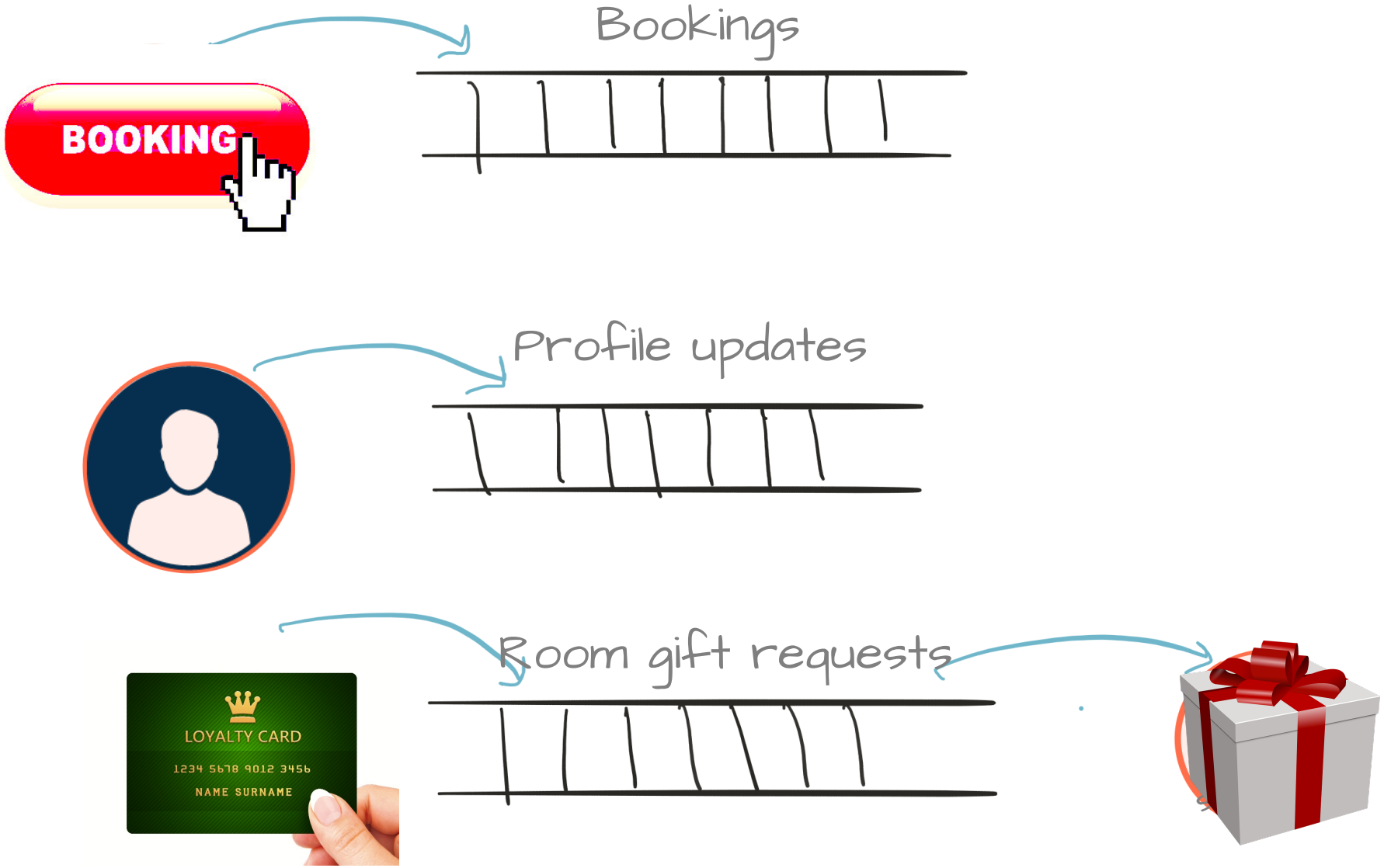
Schema are APIs.

We need specifications
We need to make changes to them
We need to detect breaking changes
We need versions
We need tools

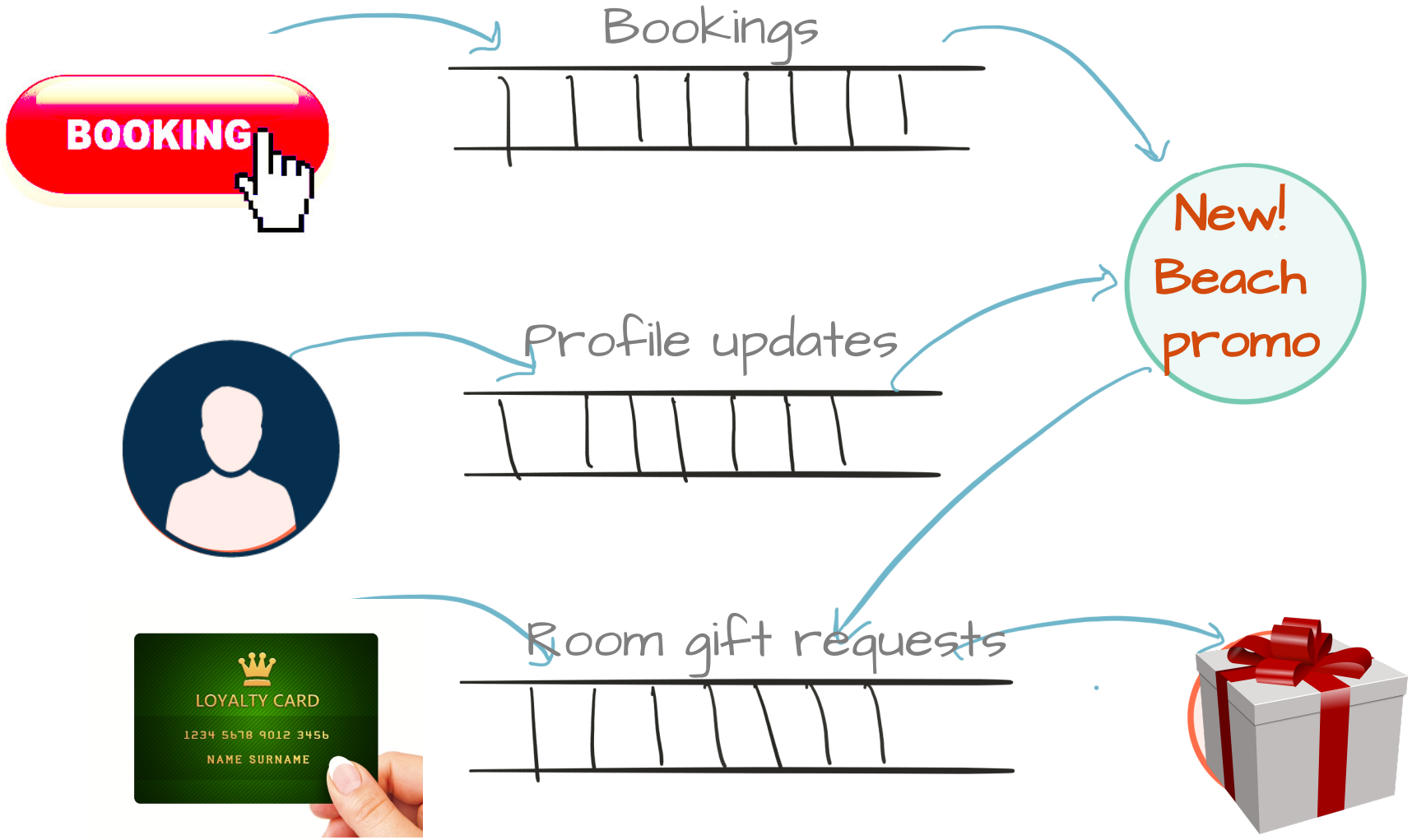
Imagine a world
where engineers
can find
the data they need
and use it safely.

Its easy if you try

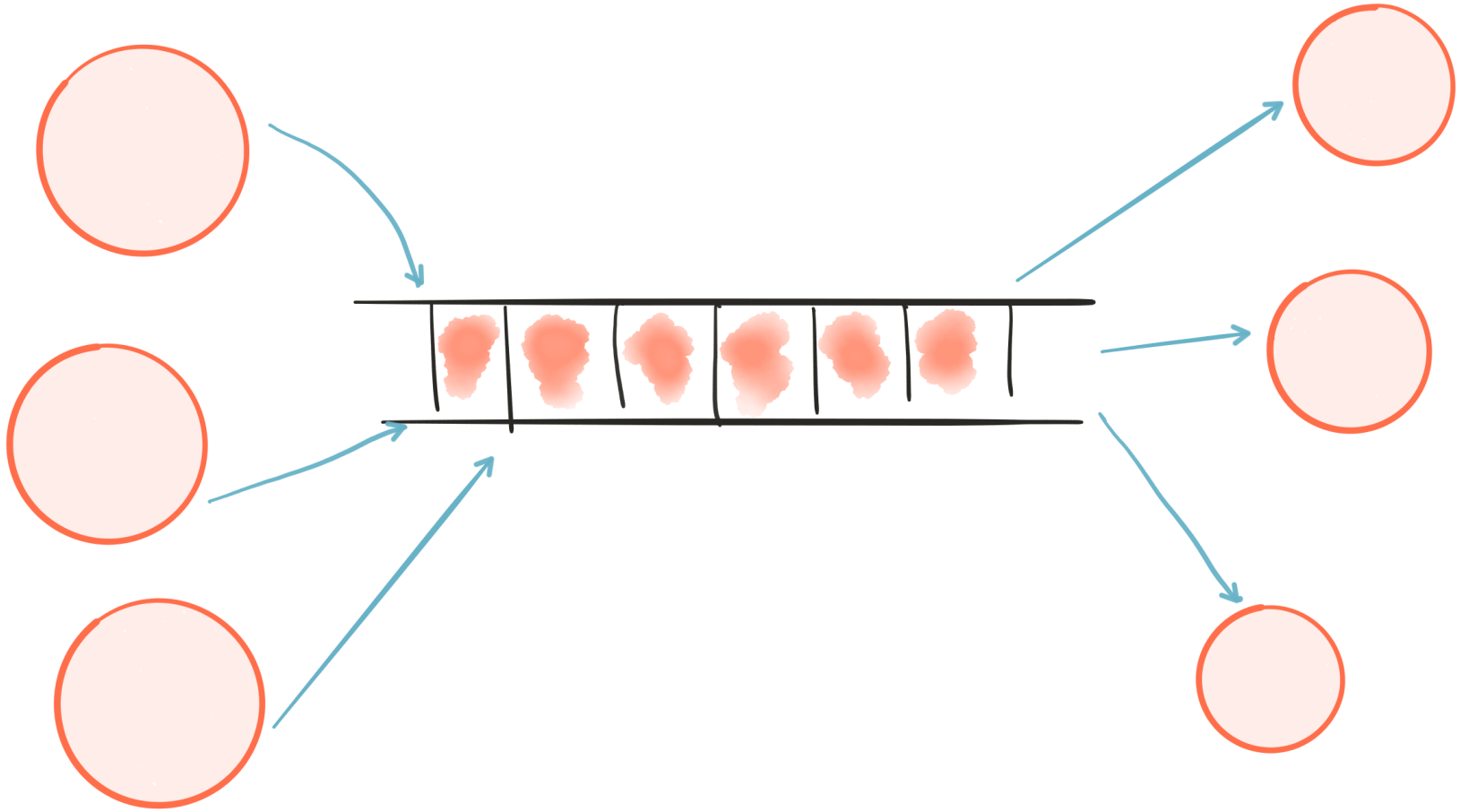
There are benefits to doing this well



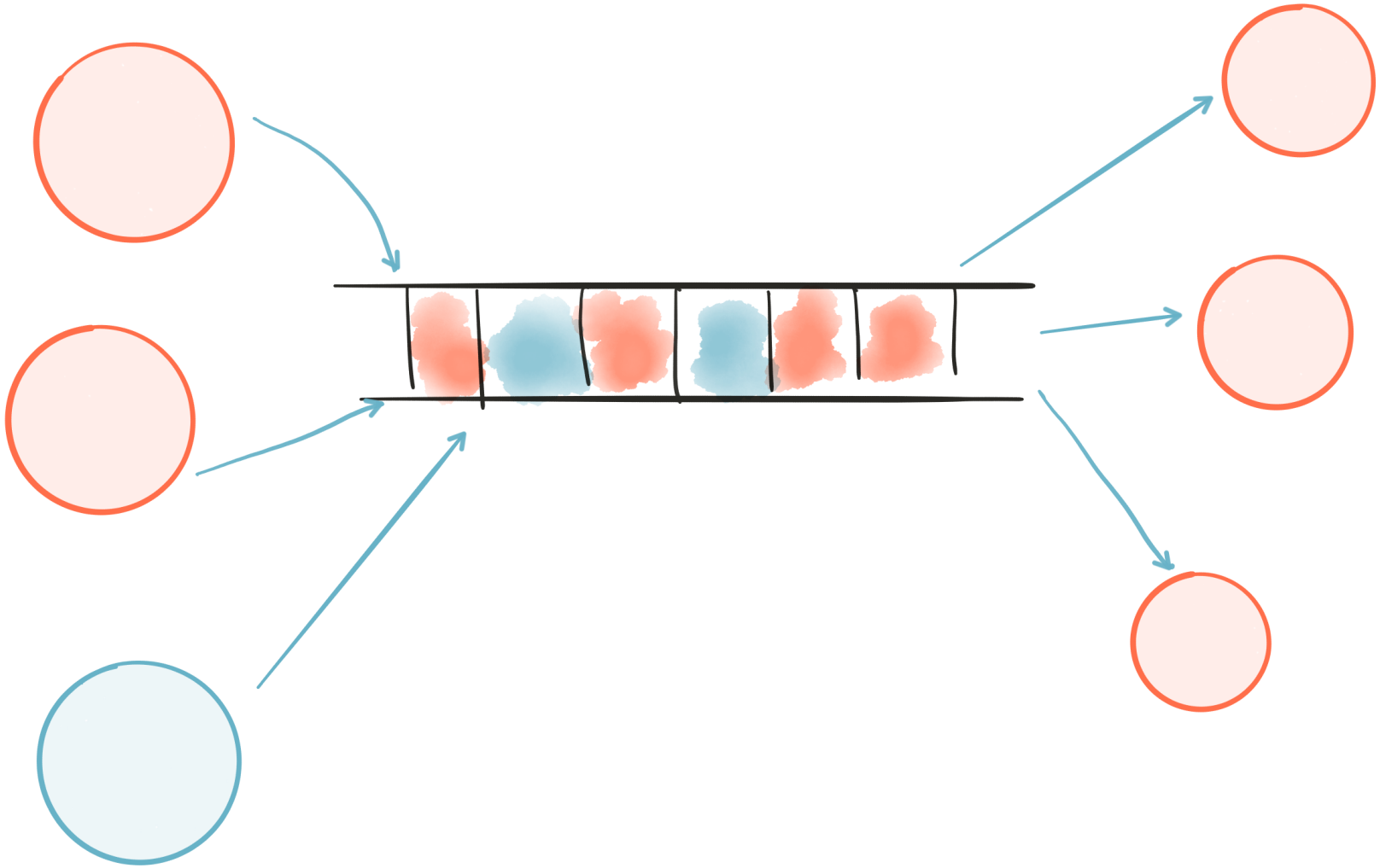
Sometimes, magic happens



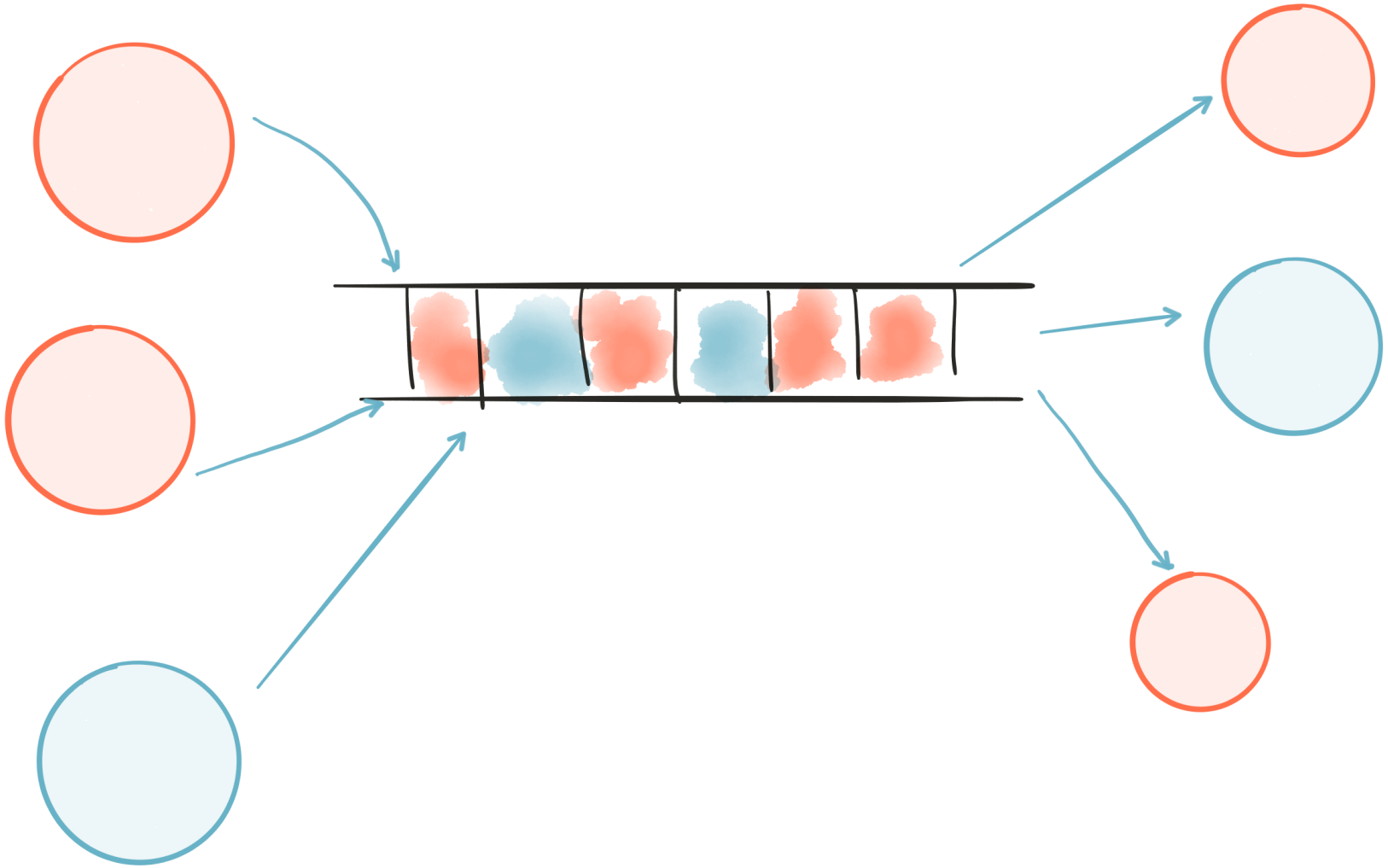
… but most days I'm happy if the data pipelines are humming and nothing breaks.



Forward compatibility:



Forward & Backward compatibility:



Compatibility Rules

	Avro	JSON
Forward Compatibility	Can add fields Can delete optional fields (nullable / default)	Can add fields
Backward Compatibility	Can delete fields Can add optional fields	Can delete fields
Full Compatibility	Can only modify optional fields	Nothing is safe

It is confusing. So it is tempting to simplify

"Never change anything"

"Adding fields is ok. Deleting is not"

"Everything is always optional except for the primary key"

Enter Schema Registry

Avro / AVRO-1124

RESTful service for holding schemas

Comment | Agile Board | More | Export

Details

Type:	New Feature	Status:	OPEN
Priority:	Major	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	None
Component/s:	None		
Labels:	None		

Description

Motivation: It is nice to be able to pass around data in serialized form but still know the exact schema that was used to serialize it. The overhead of storing the schema with each record is too high unless the individual records are very large. There are workarounds for some common cases: in the case of files a schema can be stored once with a file of many records amortizing the per-record cost, and in the case of RPC the schema can be negotiated ahead of time and used for many requests. For other uses, though it is nice to be able to pass a reference to a given schema using a small id and allow this to be looked up. Since only a small number of schemas are likely to be active for a given data source, these can easily be cached, so the number of remote lookups is very small (one per active schema version).

Basically this would consist of two things:
1. A simple REST service that stores and retrieves schemas
2. Some helper java code for fetching and caching schemas for people using the registry

People

Assignee:	Jay Kreps
Reporter:	Jay Kreps
Votes:	36 Vote for this issue
Watchers:	73 Stop watching this issue

Dates

Created:	10/Jul/12 20:46
Updated:	09/Feb/16 08:45

Agile

View on Board

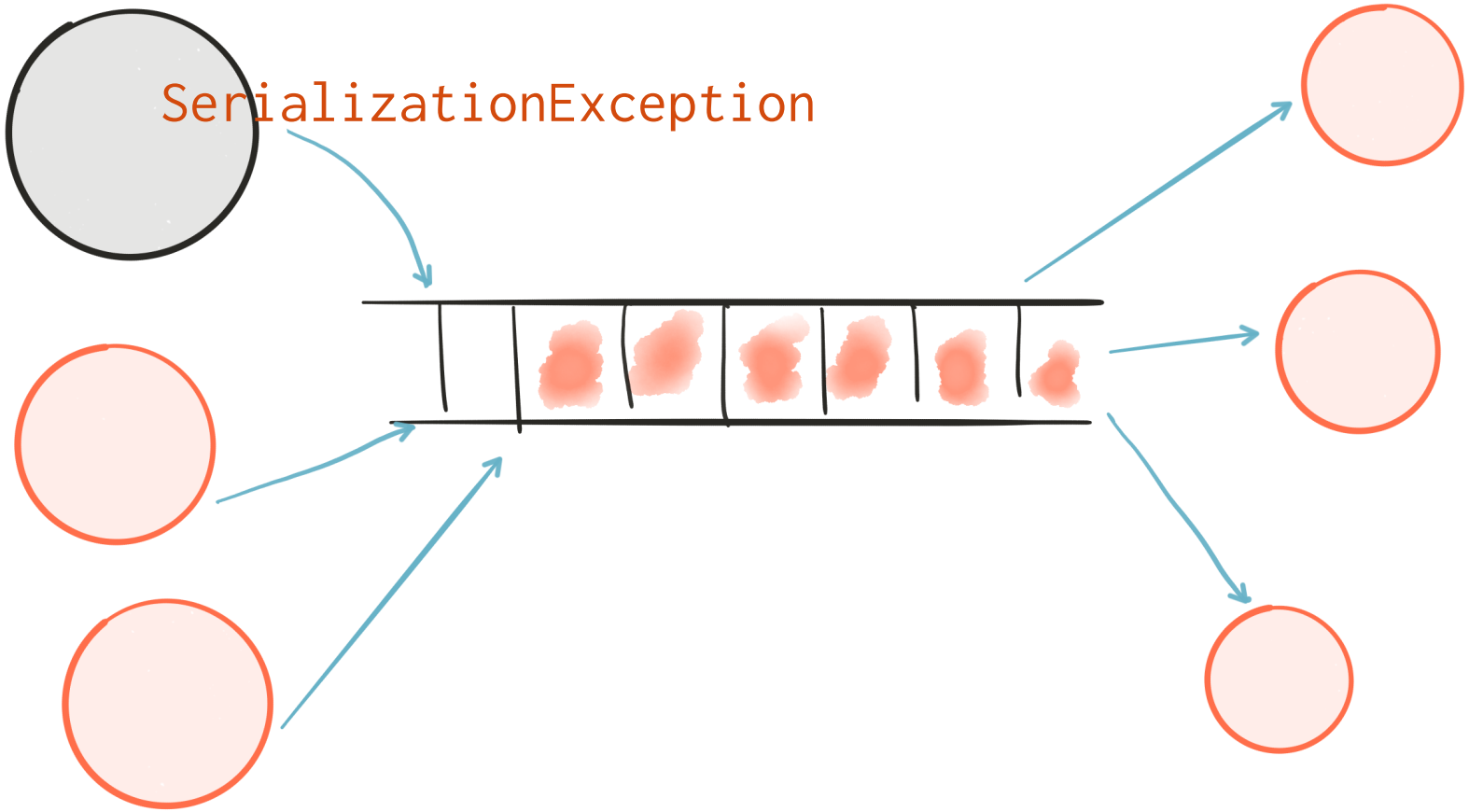
HipChat discussions

Schema Registries Everywhere

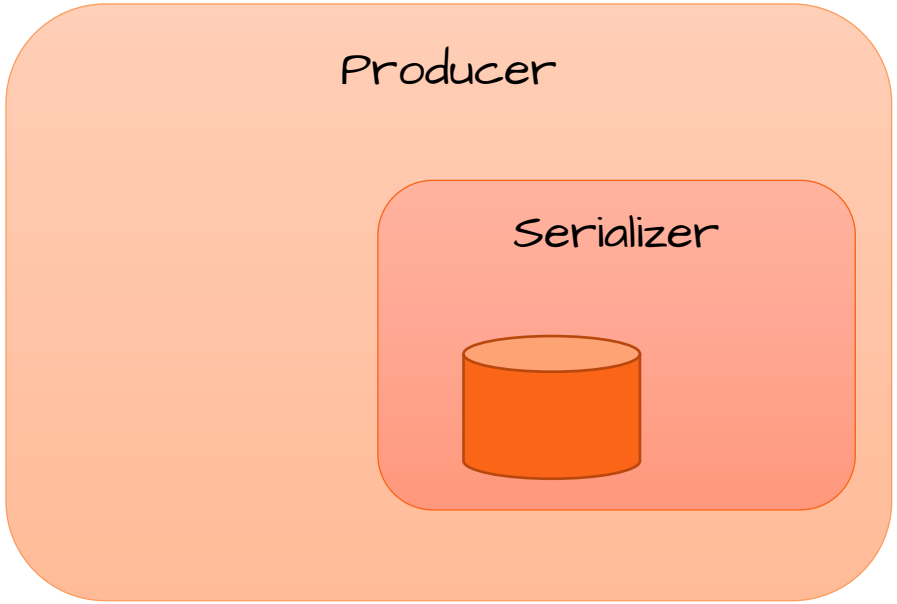
What do Schema Registries do?

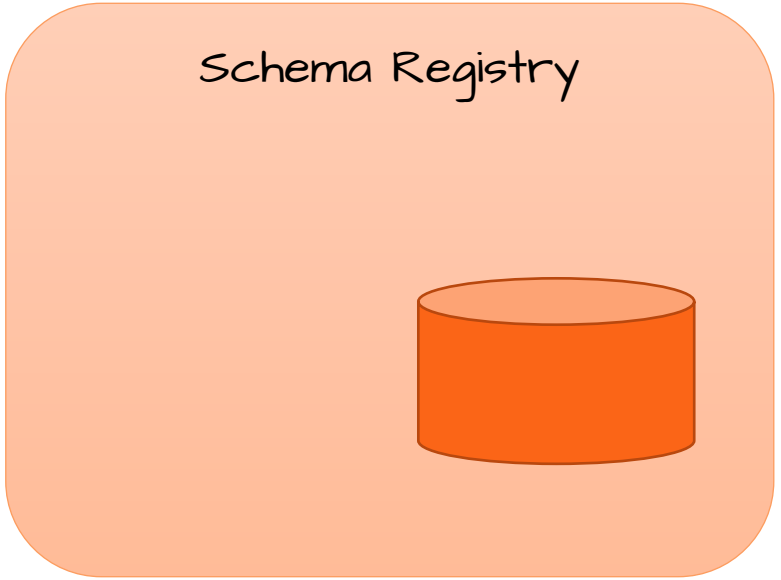
1. Store schemas - put/get
2. Link one or more schema to each event
3. Java client that fetches & caches schemas
4. Enforcement of compatibility rules
5. Graphical browser

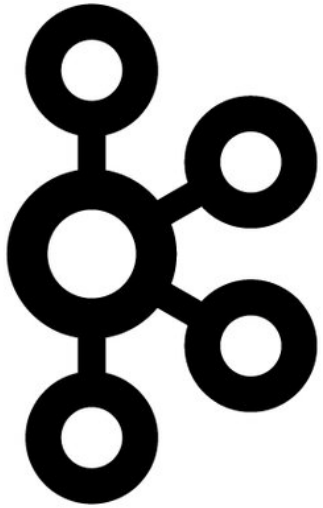
Make those contracts binding



Responsibility is slightly distributed







Producers contain Serializers

1. Define the serializers:

```
props.put("key.serializer", ”org.apache.kafka.serializers.StringSerializer");
props.put("value.serializer", "io.confluent.kafka.serializers.KafkaAvroSerializer");
props.put("schema.registry.url", schemaUrl);
…
producer<String, LogLine> producer = new KafkaProducer<String, LogLine>(props);
```

2. Create a record:

```
ProducerRecord<String, LogLine> record =
  new ProducerRecord<String, LogLine>(topic, event.getIp().toString(), event);
```

3. Send the record:

```
producer.send(record);
```

Serializers cache schemas, register new schema … and serialize

```
serialize(topic, isKey, object):
    subject = getSubjectName(topic, isKey)
    schema = getSchema(record)
    schemaIdMap = schemaCache.get(subject)
    if (schemaIdMap.containsKey(schema):
        id = schemaIdMap.get(schema)
    else
        id = registerAndGetId(subject, schema)
    schemaIdMap.put(schema, id)
    output = MAGIC_BYTE + id + avroWriter(schema, object)
```

Schema Registry caches schemas and validates compatibility

```
register(schema, subject):
    if (schemaIsNewToSubject):
        prevSchema = getPrevSchema(subject)
        level = getCompatibilityLevel(subject)
        if (level == FULL):
            validator =
                new SchemaValidatorBuilder().mutualReadStrategy().validateLatest()
            if (validator.isCompatible(schema, prevSchema))
                register
            else
                throw
        …
```

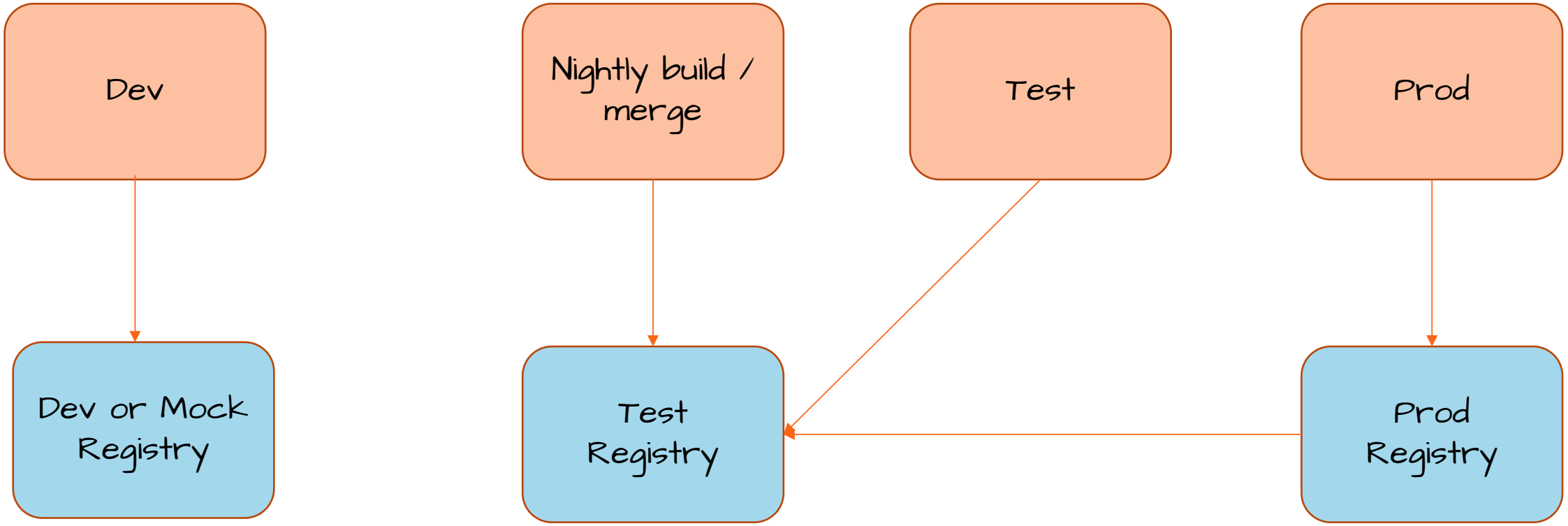

I DONT ALWAYS VALIDATE COMPATIBILITY
BUT WHEN I DO, IT'S IN PRODUCTION
imgflip.com

Maven Plugin – because we prefer to catch problems in CI/CD

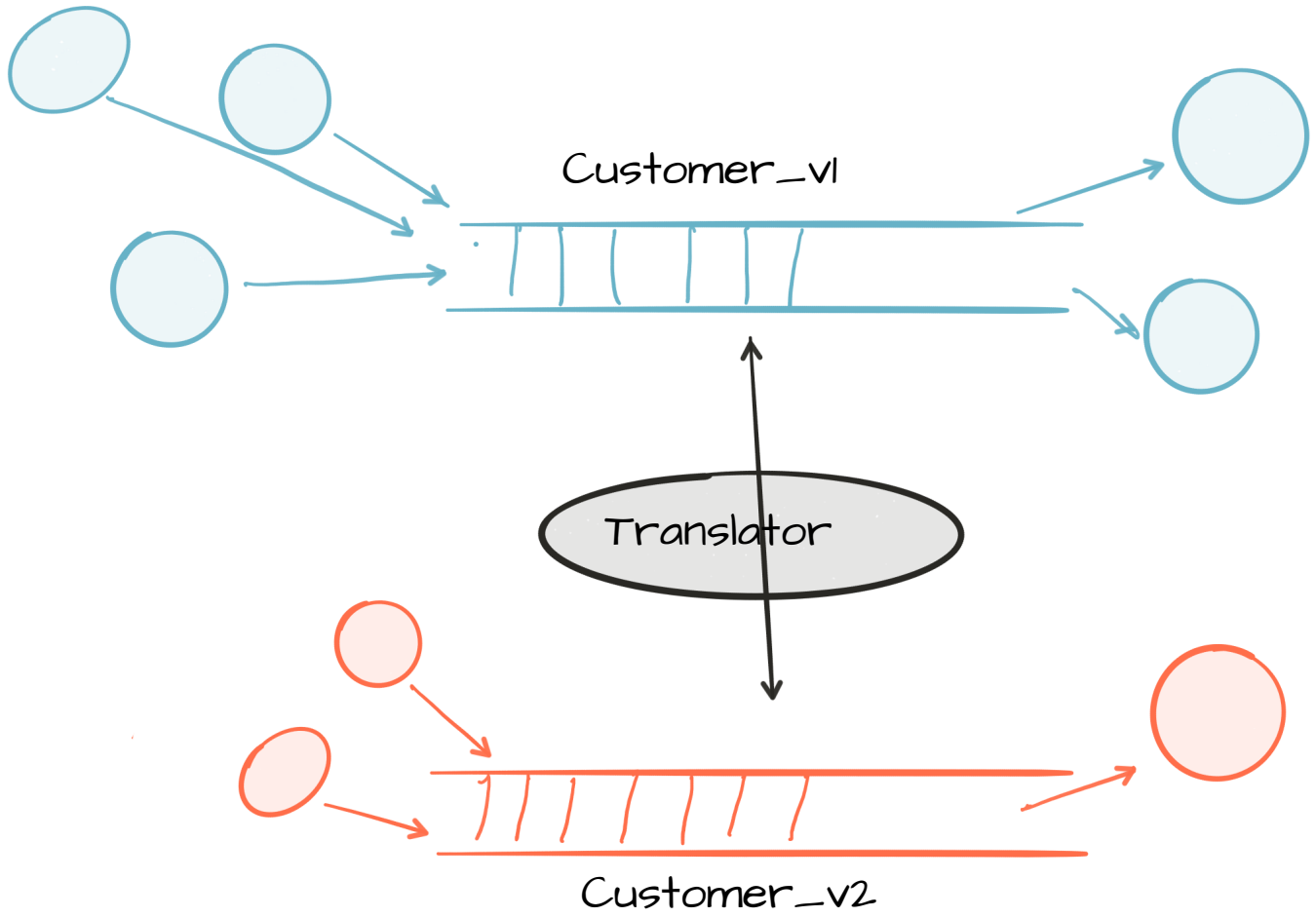
http://docs.confluent.io/current/schema-registry/docs/maven-plugin.html

- schema-registry:download
- **schema-registry:test-compatibility**
- schema-registry:register

So the flow is…

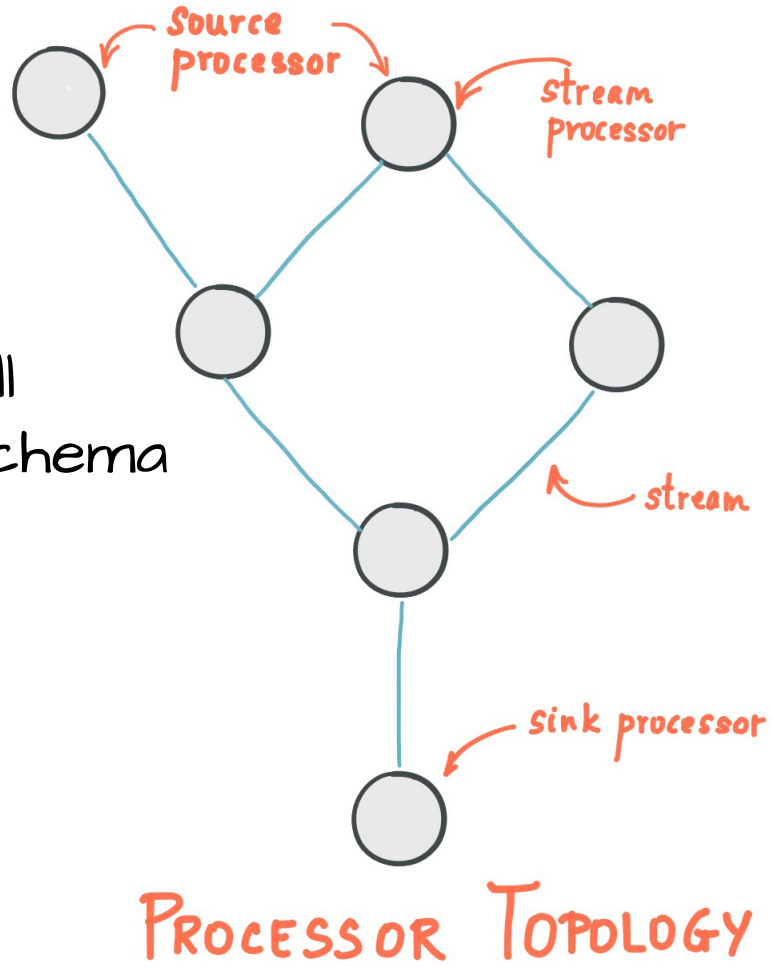


What if…. I NEED to break compatibility?



I have this stream processing job…

Nodes ~~can~~ will modify the schema



Tracking services for fun and profit

Schema discovery for fun and profit

Can we enforce compliance better?

Speaking of headers...

And really, as an old school DBA
I miss my constraints

Why should Avro users have all the fun?

Summary!

1. Schema are APIs for event-driven services
2. Which means compatibility is critical
3. Use Schema Registry from Dev to Prod
4. Schema Registry is in Confluent Open Source

Thank You!