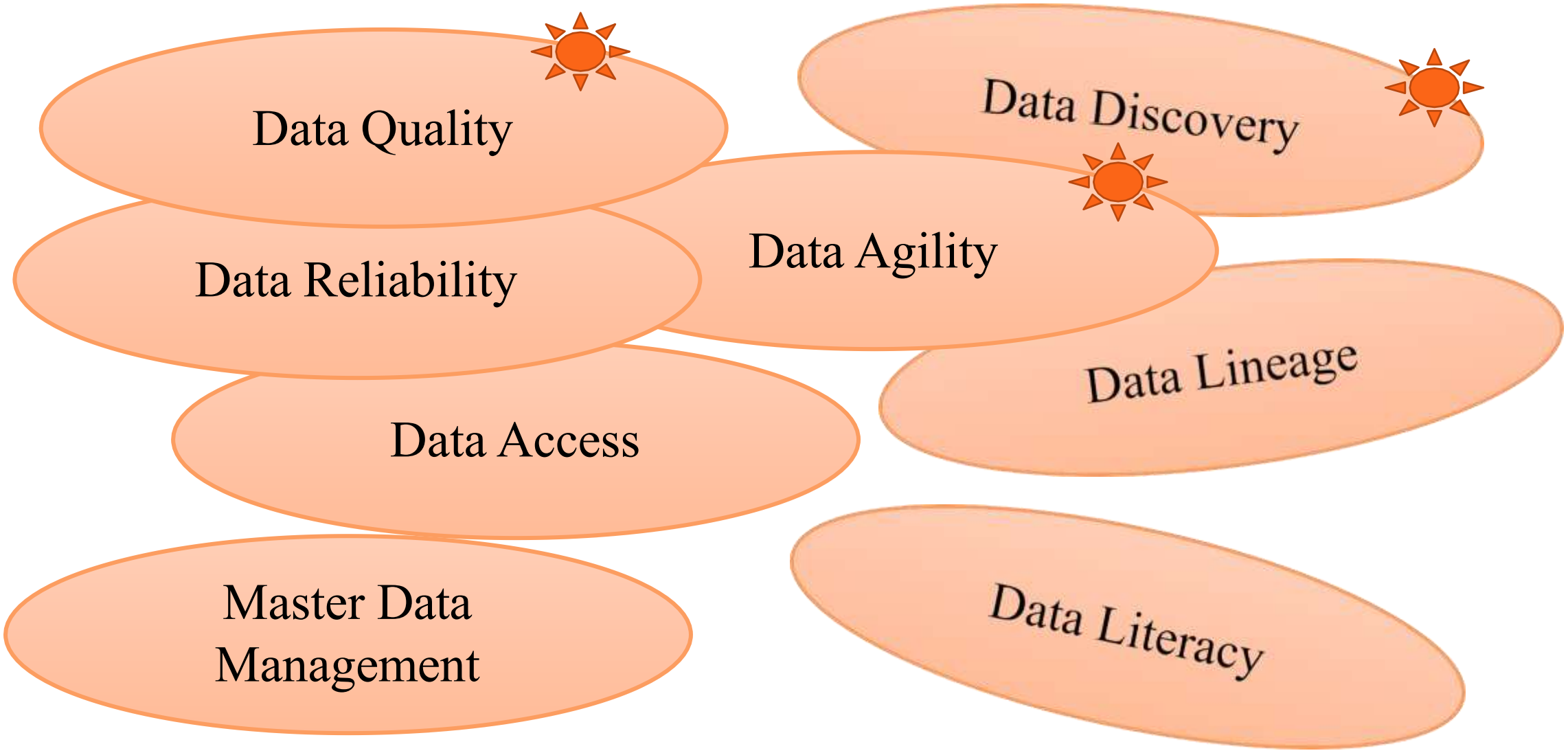
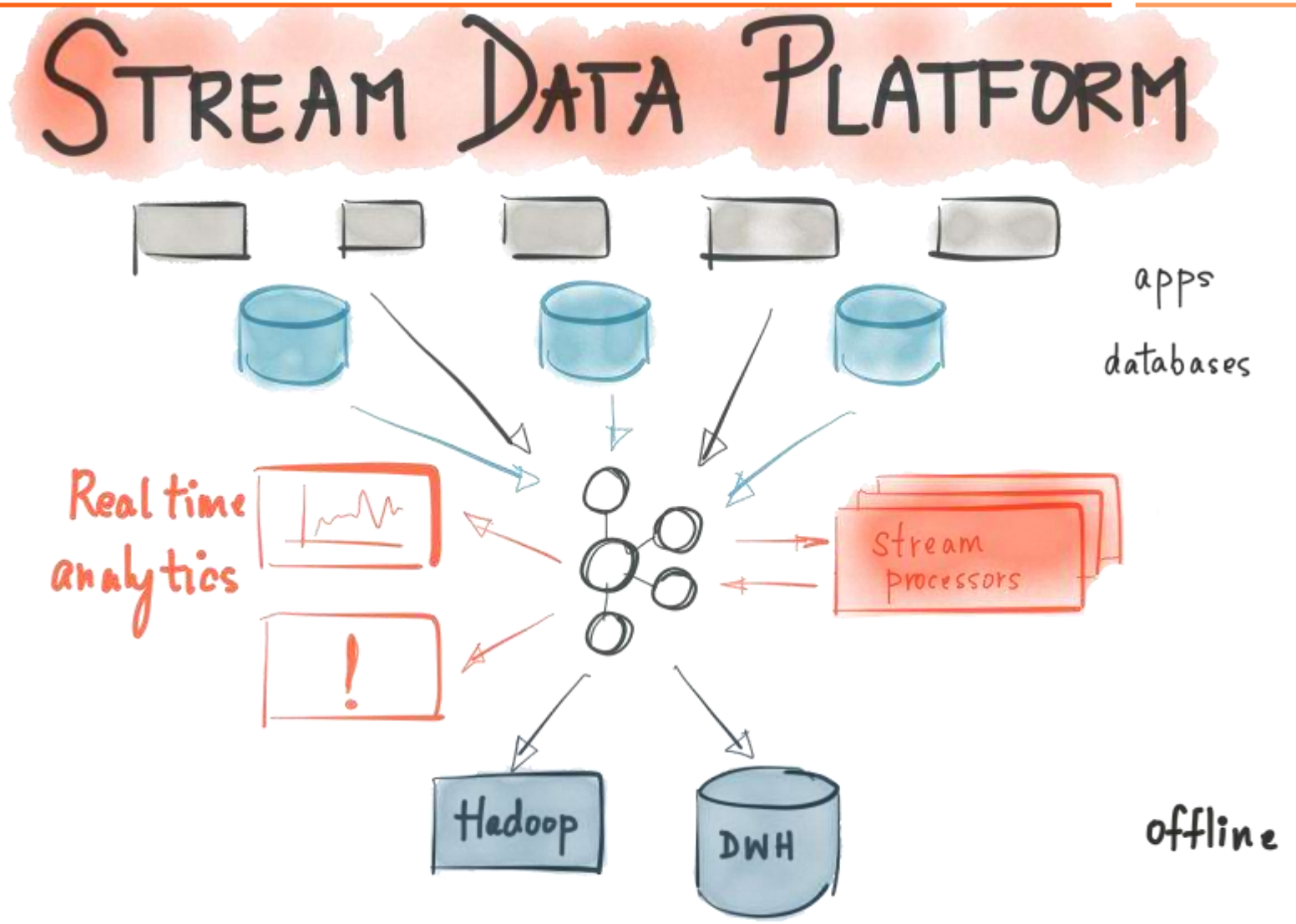
Simplify Governance of Streaming Data

What is Data Governance?



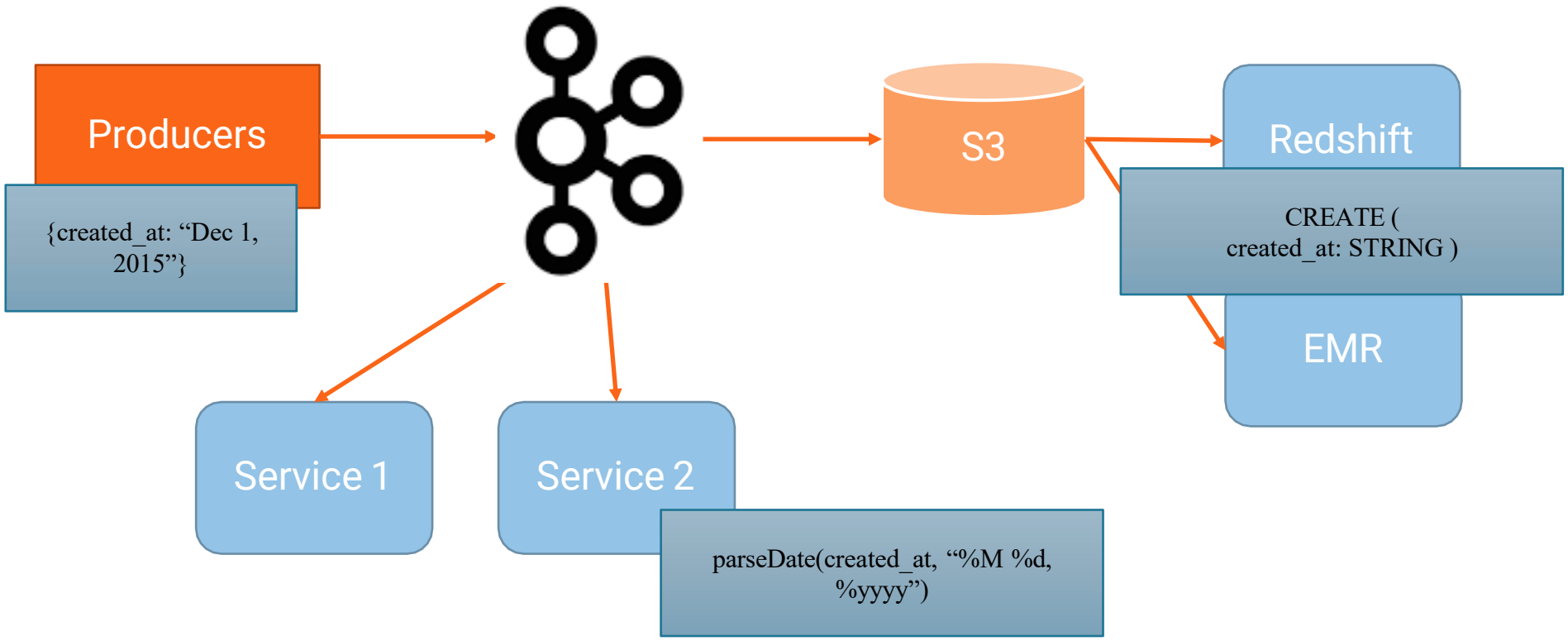
To Grow a Successful Data Platform, You need Some Guarantees



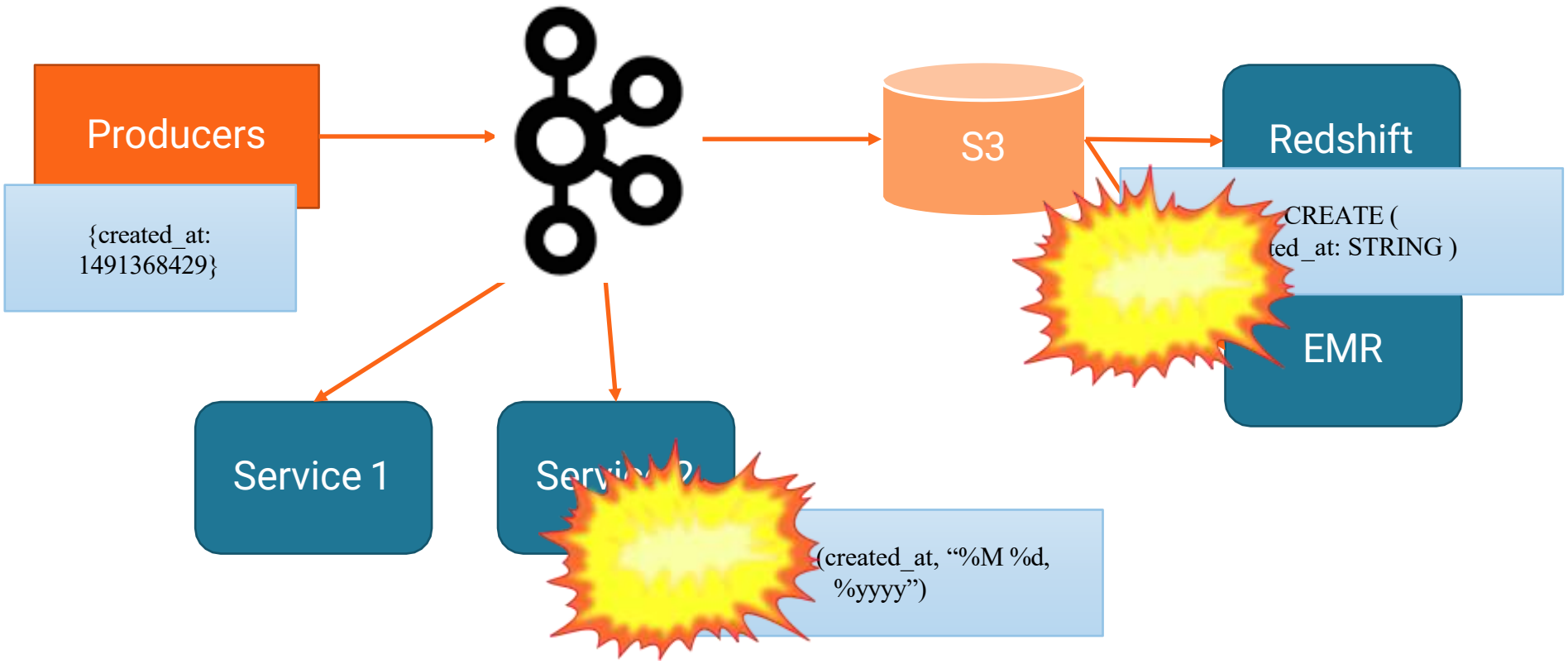
Schemas Are a Key Part of This

1. Schemas: Enforced Contracts Between your Components
2. Schemas need to be agile but compatible
3. You need tools & process that will let you:
   - Move fast and not break things

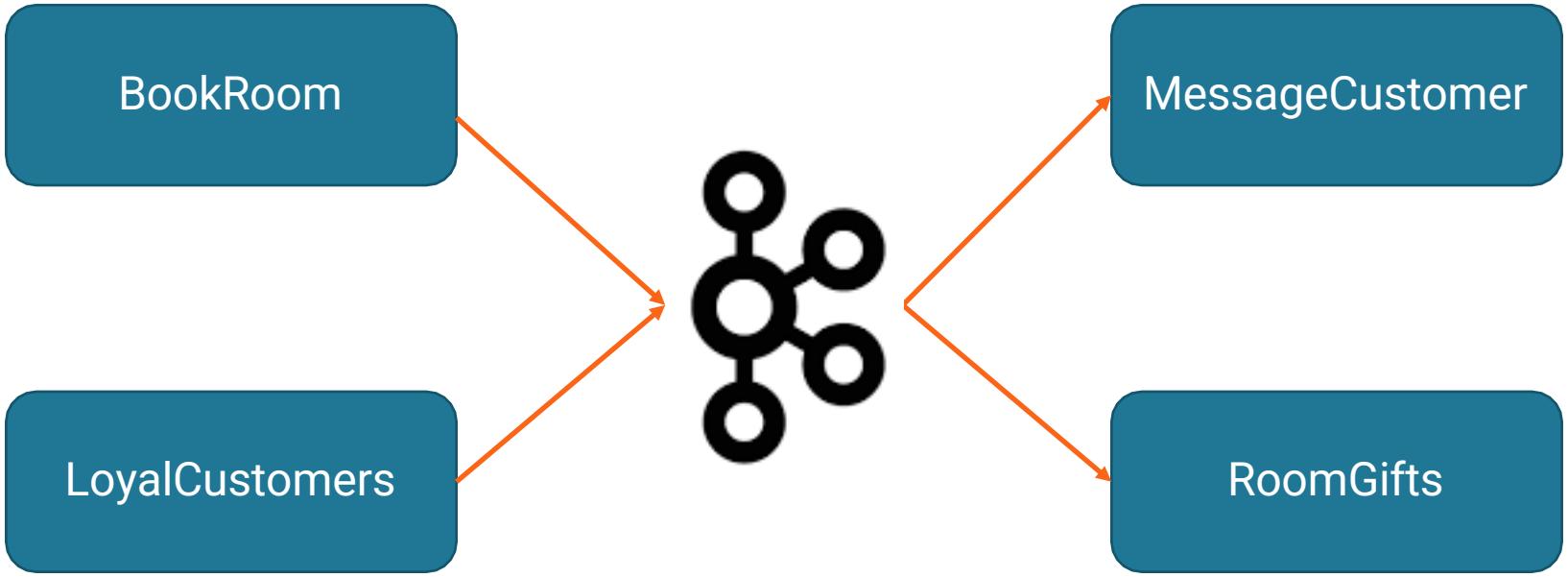
Cautionary tale: Uber



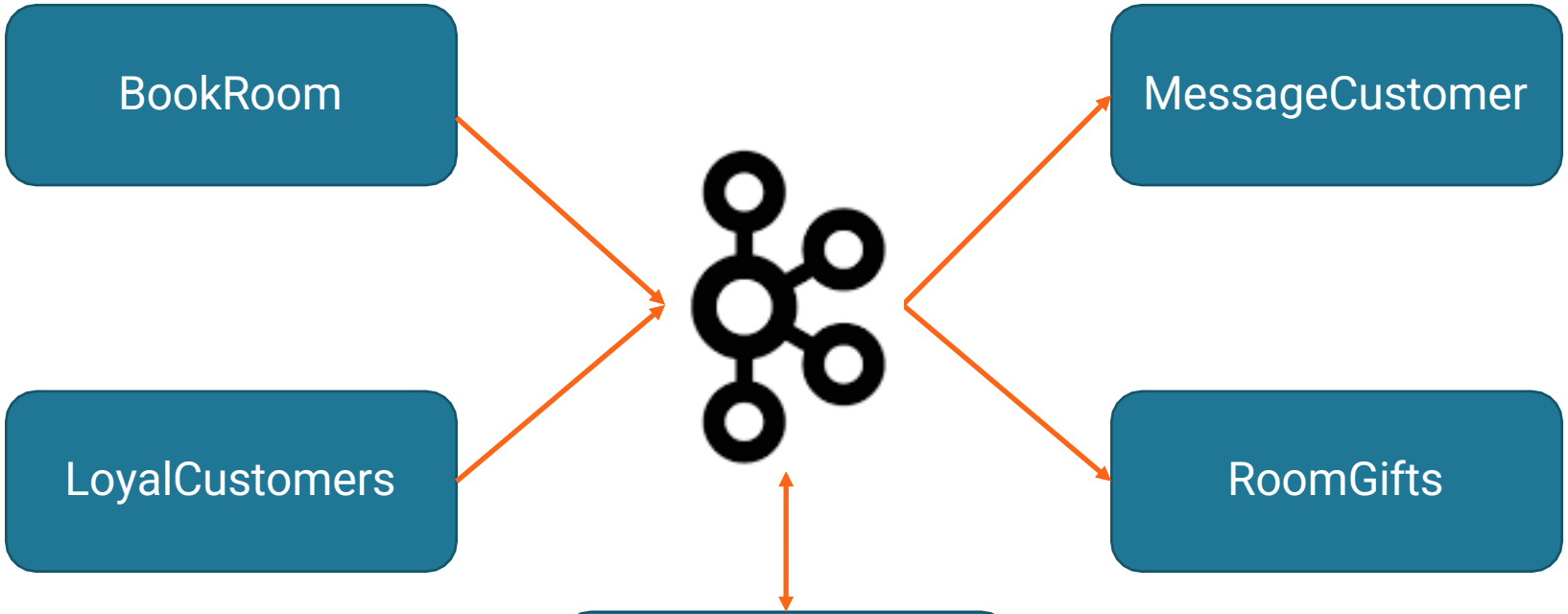
Cautionary tale: Uber



The right way to do things



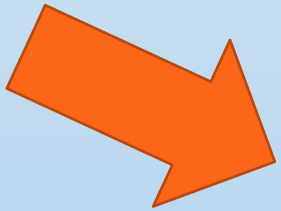
The right way to do things

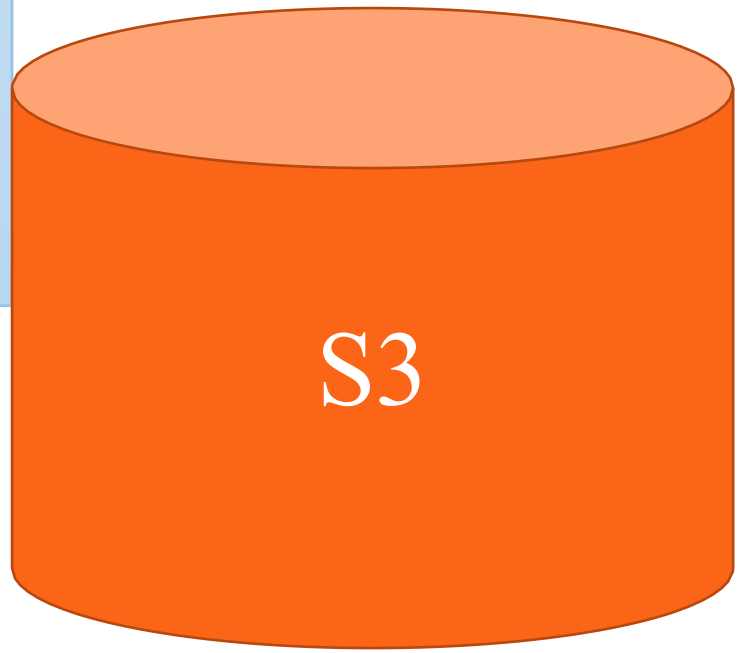



```
If (property.location == beach &&
customer.level == platinum)
sendMessage({topic: RoomGifts, customer_id: customer.id, room:
property.roomnum, gift: seaShell})
```

There is special value in standardizing parts of schema

```
{ "type": "record",
  "name": "Event",
  "fields": [{ "name": "headers", "type": {
                        "name" : "headers",
                        "type": "record"
                        "fields": [
                                {"name": "source_app", "type": "string"},
                                {"name": "host_app", "type": "string"},
                                {"name": "destination_app", "type": ["null","string"]},
                                {"name": "timestamp", "type": "long"}
                        ]},
                { "name": "body", "type": "bytes" }] }
```





Standard headers allowed
analyzing patterns of communication

In **Request Response** World – **APIs between services** are key

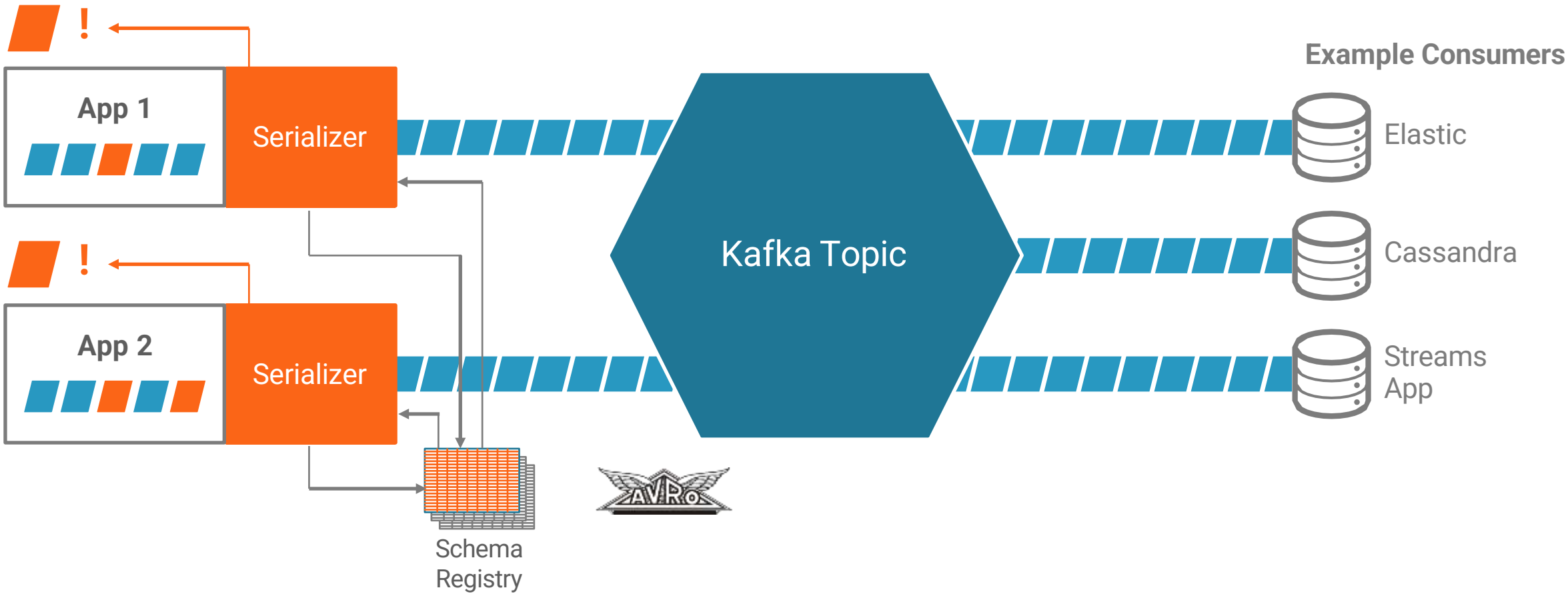
In **Async Stream Processing** World – **Message Schemas** ARE the API

…except they stick around for a lot longer

Lets assume you see why you need schema compatibility…

How do we make it work for us?

Schema Registry



Define the expected fields for each Kafka topic

Automatically **handle** schema changes (e.g. new fields)

Prevent incompatible changes

Supports multi-datacenter environments

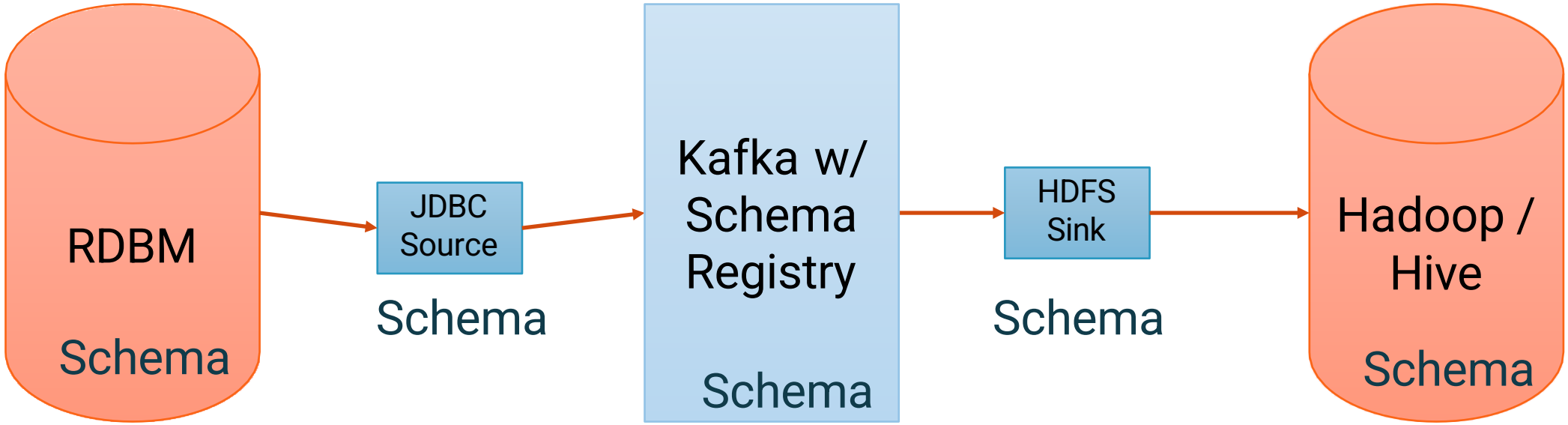
Key Points:

- Schema Registry is efficient – due to caching
- Schema Registry is transparent
- Schema Registry is Highly Available
- Schema Registry prevents bad data at the source

Big Benefits

- Schema Management is part of your entire app lifecycle – from dev to prod
- Single source of truth for all apps and data stores
- No more “Bad Data” randomly breaking things
- Increased agility! add, remove and modify fields with confidence
- Teams can share and discover data
- Smaller messages on the wire and in storage

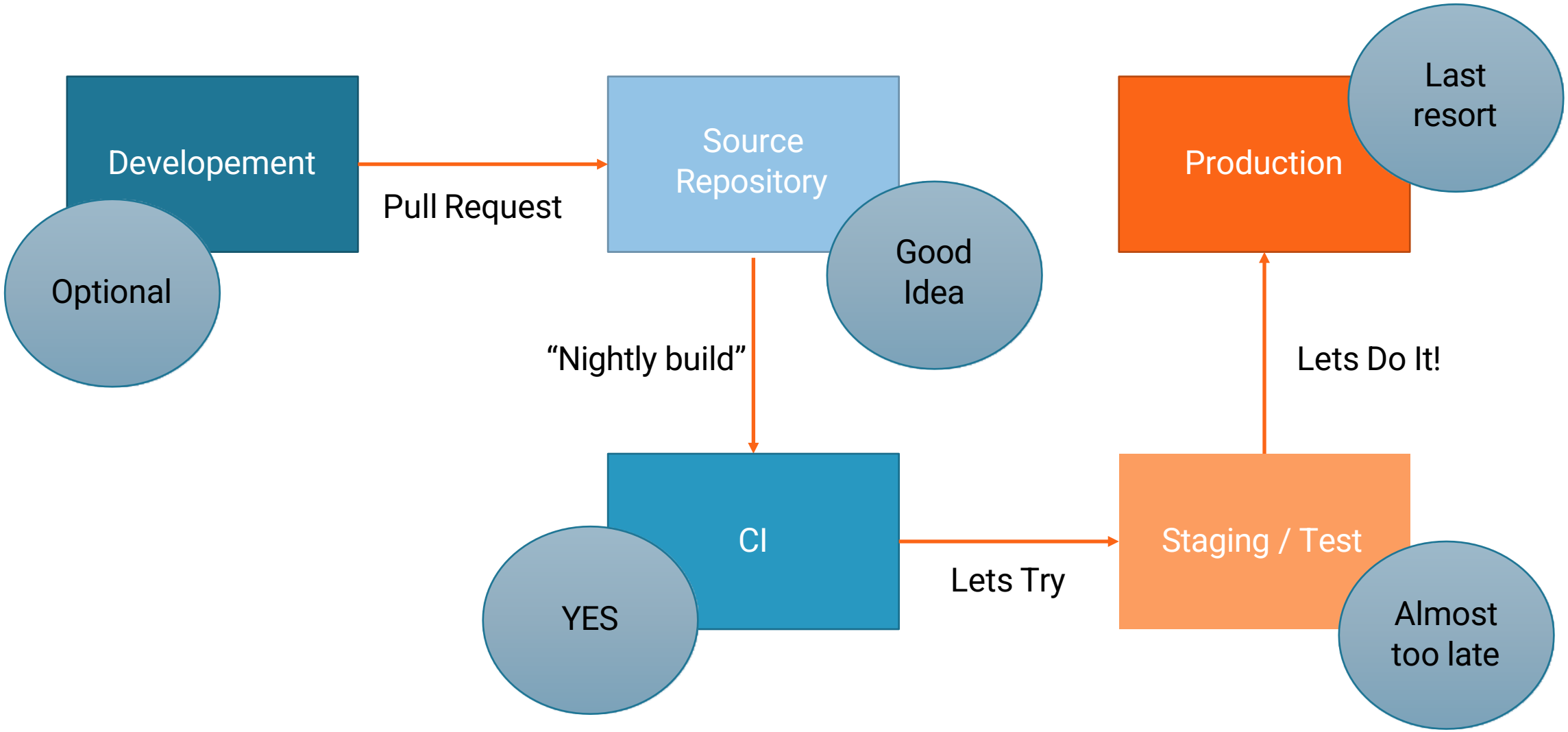
But the Best Part is…



Compatibility tips!

- **Do you want to upgrade producer without touching consumers?**
  - You want *forward* compatibility.
  - You can add fields.
  - You can delete fields with defaults
- **Do you want to update schema in a DWH but still read old messages?**
  - You want *backward* compatibility
  - You can delete fields
  - You can add fields with defaults
- **Both?**
  - You want *full* compatibility.
  - Default all things!
  - Except the primary key which you never touch.

App Lifecycle



Finding Schemas

1. Write Schemas -> Generate Code
2. Get Schemas from registry -> Generate code
3. Write code -> Generate Schemas

Summary!

1. Data Quality is critical
2. Schema are critical
3. Schema Registry is used to maintain quality from Dev to Prod
4. Schema Registry is in Confluent Open Source