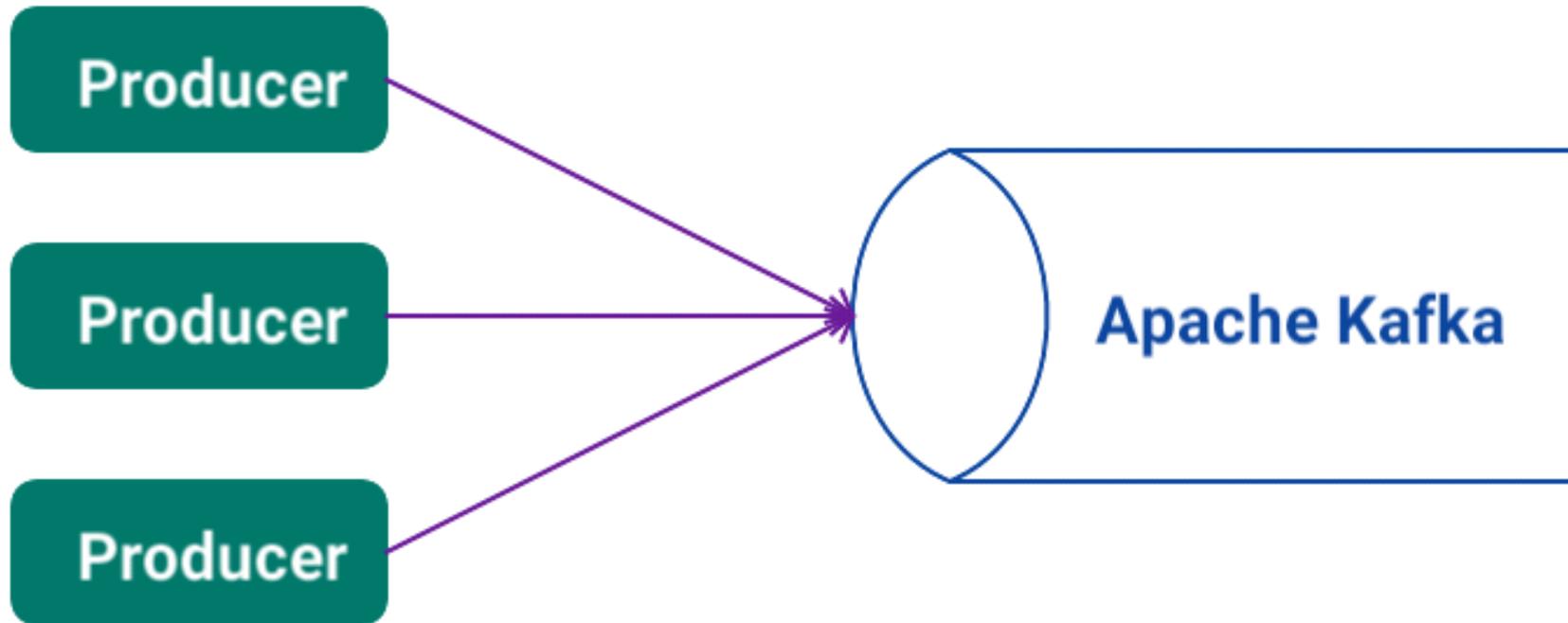


Designing Payloads for Event-Driven Systems

Lorna Mitchell, Aiven



Event-Driven Systems



Apache Kafka

"Apache Kafka is an open-source distributed event streaming platform" - <https://kafka.apache.org>

- Storage designed for data streaming
- Messages are sent to "Topics"
- Data can be ... anything



Payloads

The messages the machines send between themselves

- Large string/binary data
- No rules
- (but I do have advice!)

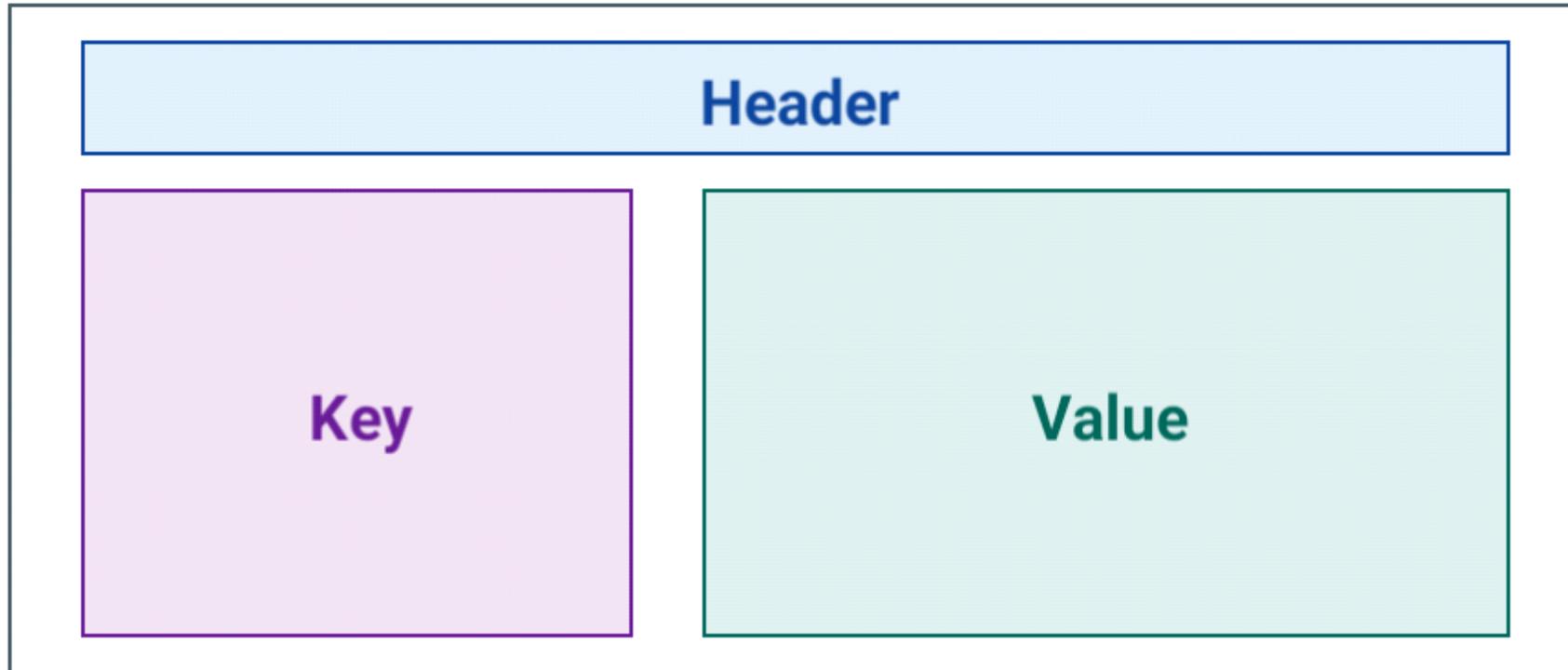


The background consists of a collage of various envelopes and tags. There are several white envelopes of different sizes and orientations, some with their flaps folded. Interspersed among them are several yellow envelopes, including a large one in the lower-left and a smaller one in the lower-right. Additionally, there are several light-colored, rectangular tags with a small hole at the top, scattered across the composition. The overall aesthetic is clean and organized, suggesting a focus on design and communication.

Payload Design Tips

Apache Kafka Records

Use all the features of Apache Kafka's records



Header

- Metadata *about* the main payload
- Available without deserializing

```
source_type: sensor  
trace_id: 1b15c98e-a52a-443d
```



Key

- Key usually sets the partition
- Can include multiple fields

```
{  
  "type": "sensor_reading",  
  "factory_id": 44891  
}
```



Flat or Nested Structures?

- Always use a top level object structure (not an array)
- Group related fields together

```
{  
  "stores_request_id": 10004352789,  
  "parent_order": {  
    "order_ref": 777289,  
    "agent": "Mr Thing (1185)"  
  },  
  "bom": [  
    {"part": "hinge_cup_sg7", "quantity": 18},  
    {"part": "worktop_kit_sm", "quantity": 1},  
    {"part": "softcls_norm2", "quantity": 9}  
  ]  
}
```



More Data or Less Data?

- For small payloads, add the context fields
- Use lightweight representation rather than the full object
- Be careful of triggering many extra lookups
- Hypermedia can help



Example: GitHub Webhooks

(snippet from the push webhook)

```
"user": {
  "login": "Codertocat",
  "id": 21031067,
  "avatar_url": "https://avatars1.githubusercontent.com/u/21031067?v=4",
  "url": "https://api.github.com/users/Codertocat",
  "html_url": "https://github.com/Codertocat",
  "followers_url": "https://api.github.com/users/Codertocat/followers",
  "following_url": "https://api.github.com/users/Codertocat/following{/other_user}",
  "gists_url": "https://api.github.com/users/Codertocat/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/Codertocat/starred{/owner}{/repo}",
  "organizations_url": "https://api.github.com/users/Codertocat/orgs",
  "repos_url": "https://api.github.com/users/Codertocat/repos",
  "type": "User",
},
```



A Note on Timestamps

- Apache Kafka includes publish time in the header.
- Consider adding payload-level timestamps.
- Timestamps only as accurate as your clock!

Pick a standard, any standard!

1615910306 or 2021-05-11T10:58:26Z

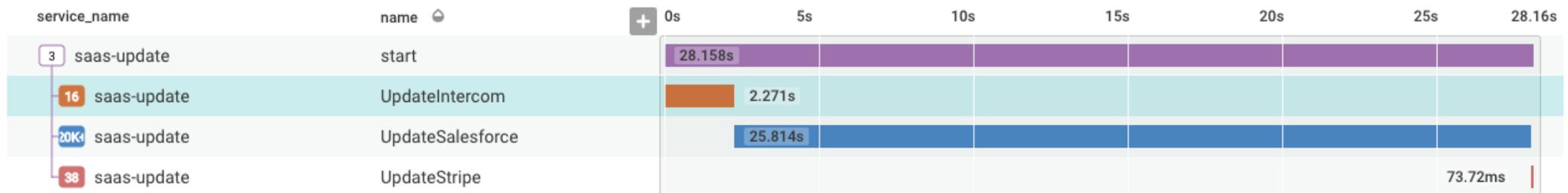


Event Tracing

Standards are great! <https://opentelemetry.io>

- Trace ID used by every event in the story
- Span ID in event, becomes Parent Span ID for child

(beautiful graph from honeycomb.io)



The background consists of a collage of various envelopes and tags. There are several white envelopes of different sizes and orientations, some with their flaps folded. Interspersed among them are several light beige or cream-colored tags, some of which have a small hole at the top, suggesting they were once attached to a string or used as labels. The overall composition is layered and textured, with the envelopes and tags overlapping each other.

Using and Evolving Schemas

Data Formats

Some formats require schemas.

- **JSON:** text-based, few data types, schema optional
- **XML:** text-based, stronger typing, schema optional
- **Language-Specific Serialization:** (it depends!)
- **Protobuf:** binary format, handled by generated code
- **Avro:** binary format, schema required



Schemas

Schemas enforce payload structure

- Avro format requires a schema
 - message has schema version information
 - used to look up fieldnames and reconstruct payload
- Schema Registry holds the schema versions for each topic



Evolving Schemas

- Aim for backwards-compatible changes
 - to rename: add the new field, keep the old one
 - safe to add optional fields
- Each change is a new version
- Avro supports aliases and default values



Example: Avro Schema

Avro schema example for sensor data

```
{
  "namespace": "io.aiven.example",
  "type": "record",
  "name": "MachineSensor",
  "fields": [
    {"name": "machine", "type": "string",
     "doc": "The machine whose sensor this is"},
    {"name": "sensor", "type": "string", "doc": "Which sensor was read"},
    {"name": "value", "type": "float", "doc": "Sensor reading"},
    {"name": "units", "type": "string", "doc": "Measurement units"}
  ]
}
```



The background consists of a collage of various envelopes and tags. There are several white envelopes of different sizes and orientations, some with their flaps folded. Interspersed among them are several light beige or cream-colored envelopes, including a prominent one in the lower-left quadrant. Additionally, there are several light-colored, rectangular tags with a small hole at the top, scattered across the composition. The overall aesthetic is clean and organized, suggesting a theme of communication or data management.

Describing Payloads



AsyncAPI for Apache Kafka

AsyncAPI describes event-driven architectures

<https://www.asyncapi.com>

We can describe the:

- brokers and auth
- topics
- payloads



Describing Payloads

The channels section of the AsyncAPI document

```
factorysensor:  
  subscribe:  
    operationId: MachineSensor  
    summary: Data from the in-machine sensors  
    bindings:  
      kafka:  
        clientId:  
          type: string  
    message:  
      name: sensor-reading  
      title: Sensor Reading  
      schemaFormat: "application/vnd.apache.avro;version=1.9.0"  
      payload:  
        $ref: machine_sensor.avsc
```



Documenting Payloads

Thingum Industries
Sensors 1.0.0

Introduction
Servers

OPERATIONS

SUB Data from the in-machine sensors

MESSAGES

sensor-reading

SUB factorysensor

Factory sensor data

Data from the in-machine sensors

Accepts the following message:

Sensor Reading `sensor-reading`

Some information from a sensor

Payload ▾ **Object**

| | |
|---------|---|
| machine | String The machine whose sensor this is |
| sensor | String Which sensor was read |
| value | Number Sensor reading |
| units | String Measurement units |

Additional properties are allowed.

Examples

sensor-reading

Payload >



The background of the slide is a collage of various white envelopes and yellow tags, some overlapping and some partially visible, creating a textured, layered effect. The text is centered over this background.

Payloads and Event-Driven Systems

Design with intention, embrace standards



Resources

- Examples: <https://github.com/aiven/thingum-industries>
- Blog post: <https://aiven.io/blog/tips-for-designing-payloads>
- Aiven: <https://aiven.io>
- Karapace: <https://karapace.io>
- AsyncAPI: <https://asyncapi.com>

