AppDeveloperCon
NORTH AMERICA

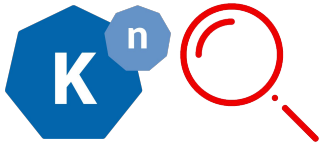
Event Discovery for Kubernetes Applications

Christoph Stäbler (@creydr)
Senior Software Engineer, Red Hat

Pierangelo Di Pilato (@pierDipi)
Principal Software Engineer, Red Hat

Agenda

Why Event Discovery?



Event Discovery in Knative Eventing

Demo and Q&A

Knative Eventing

Source

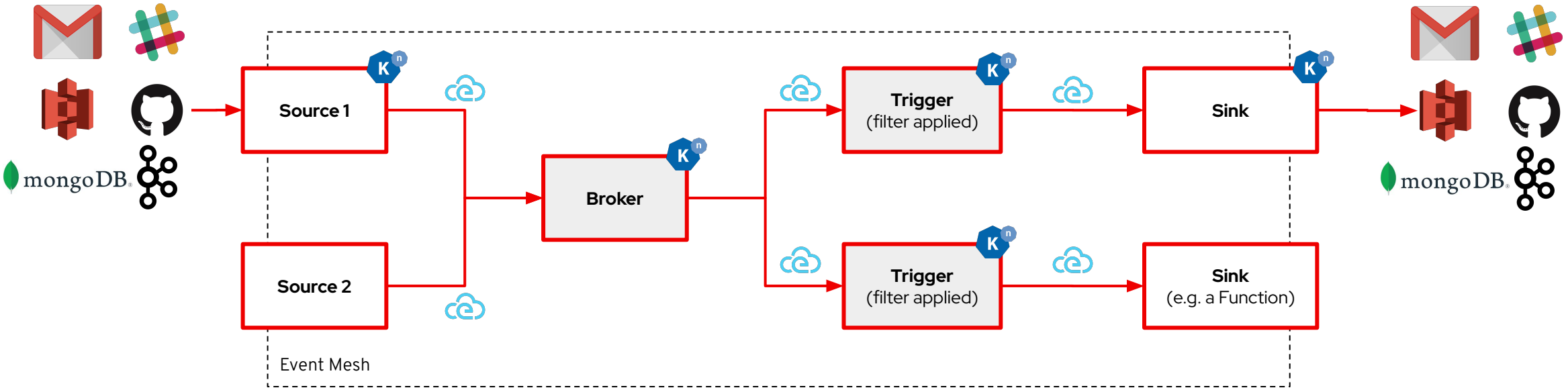
- Way to get events into the system / event-mesh
- Service in the system, which emits events

Broker & Triggers

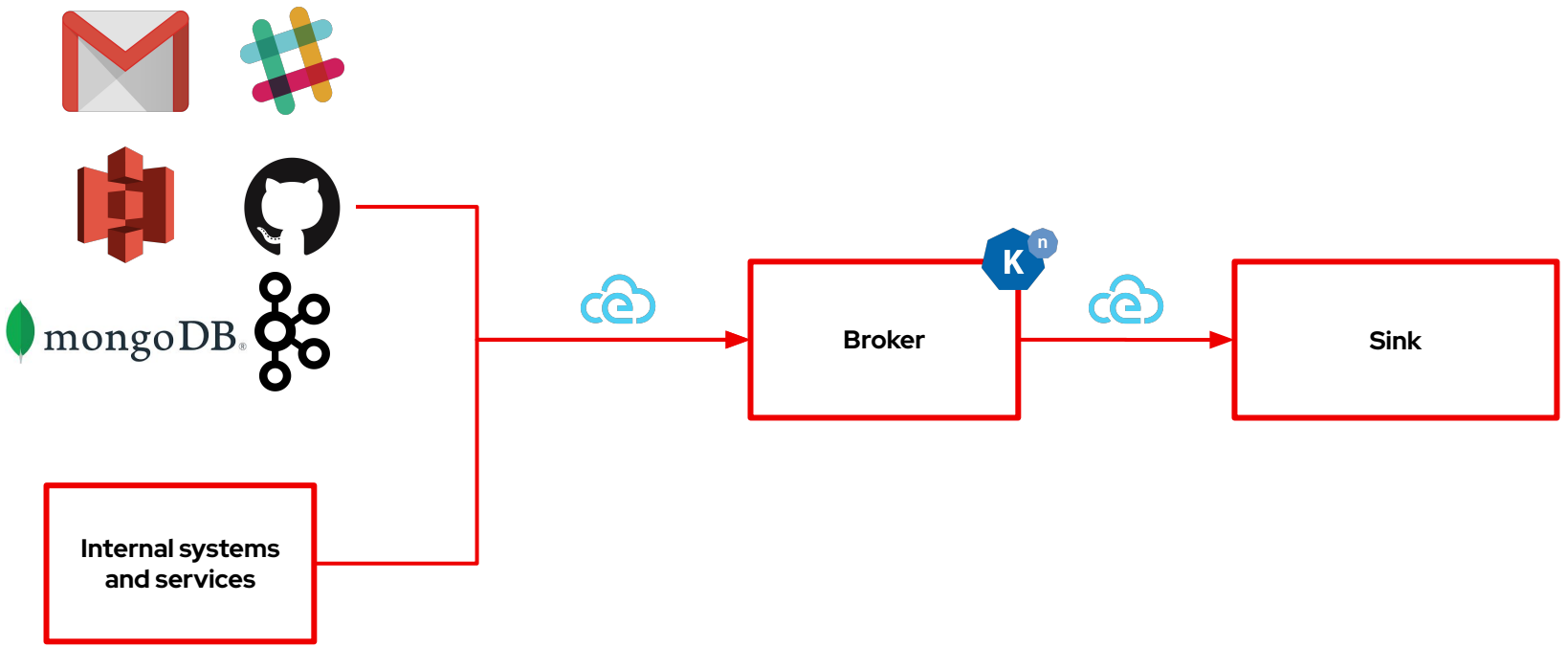
- Routing of events

Sink

- Way to get events out of the system / event-mesh to an external destination
- Service in the system, which accepts events

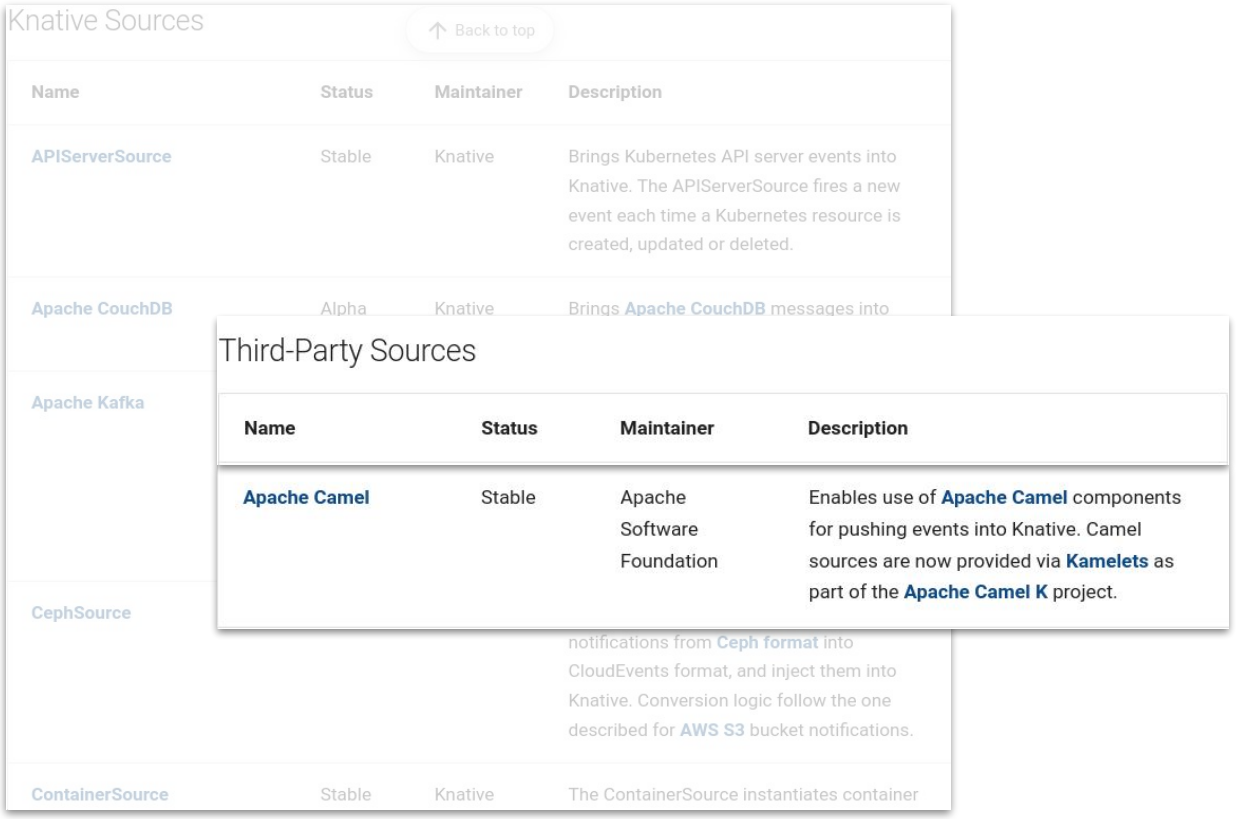


Why Event Discovery?



What are the available events I can consume?

What is their schema?



There is no information about the available types or schemas

What can I do next?

I'm using Knative Eventing with the Apache Camel (Kamelets) and I need to know the event types and schemas that I can consume from AWS DynamoDB Stream

ChatGPT

When you're using **Knative Eventing** with **Apache Camel K (Kamelets)** to consume events from AWS services like DynamoDB Streams, the Kamelets handle the integration between the Knative event-driven architecture and the external system, such as DynamoDB.

For AWS DynamoDB Streams, which captures data modifications in your DynamoDB tables, the event types and schema depend on how the stream is configured and the data it captures (e.g., "New image" or "Old image" of the modified items).

Here's an overview of the event types and schemas you can expect when consuming DynamoDB Streams using Knative Eventing with Camel K:

Event Types

Knative Eventing uses **CloudEvents** to wrap the event payloads coming from external systems like DynamoDB Streams. The CloudEvent will include certain metadata, including the type of the event.

For DynamoDB Streams, you'll typically encounter these event types:

1. `aws.dynamodb.insert` : An item was inserted into the DynamoDB table.
2. `aws.dynamodb.modify` : An existing item was modified in the DynamoDB table.
3. `aws.dynamodb.remove` : An item was deleted from the DynamoDB table.

These types directly map to DynamoDB Streams record types:

- **INSERT**: Represents a new item added to the table.
- **MODIFY**: Represents a modified item.
- **REMOVE**: Represents a deleted item.

ChatGPT

Example Event Schema

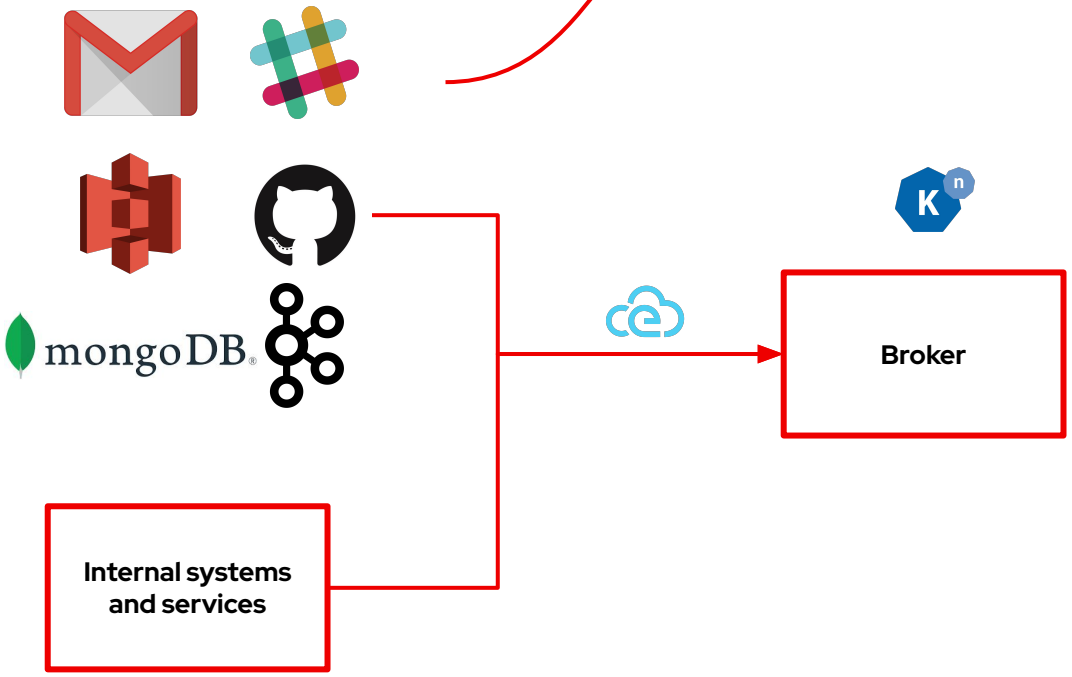
For an `INSERT` event, the CloudEvent `data` payload might look like this:

json Copy code

```
{
  "eventID": "1",
  "eventName": "INSERT",
  "eventVersion": "1.1",
  "eventSource": "aws:dynamodb",
  "awsRegion": "us-east-1",
  "dynamodb": {
    "Keys": {
      "PrimaryKey": {
        "S": "PrimaryKeyValue"
      }
    },
    "NewImage": {
      "AttributeName1": {
        "S": "AttributeValue1"
      },
      "AttributeName2": {
        "N": "123"
      }
    },
    "StreamViewType": "NEW_AND_OLD_IMAGES",
    "SequenceNumber": "111",
    "SizeBytes": 26
  }
}
```

ChatGPT

Let's assume we trust ChatGPT to describe public sources well



What about the internal systems and services?

Trial and error is the only solution today

```
function handler(event) {
  console.log(event.data)
}
```

```
$ oc logs -l=app=my-application -f

{ "Records":[{"eventID":"78dfd1ba7a17", "eventName":"INSERT", "eventVersion":"1.1",
"eventSource":"aws:dynamodb", "awsRegion":"ap-southeast-2", "dynamodb":[{"id":{"S":"xxx"},
"type":{ "S":"xxx"}}, "NewImage":{ }, "OldImage":{ } ]}]}
```

Event Discovery in Knative Eventing

EventType API

```yaml
apiVersion: eventing.knative.dev/v1beta3
kind: EventType
metadata:
  ...
spec:
  reference:
    apiVersion: camel.apache.org/v1
    kind: Kamelet
    name: github-event-source
  attributes:
    - name: type
      value: com.github.pull_request.opened
    - name: source
      value: "https://github.com/cloudevents/spec/pull/{id}"
    - name: dataschema
      value: "https:// ... "
```

Reference to the emitting resource

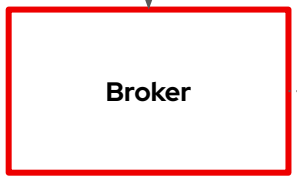
Attributes of the emitted event

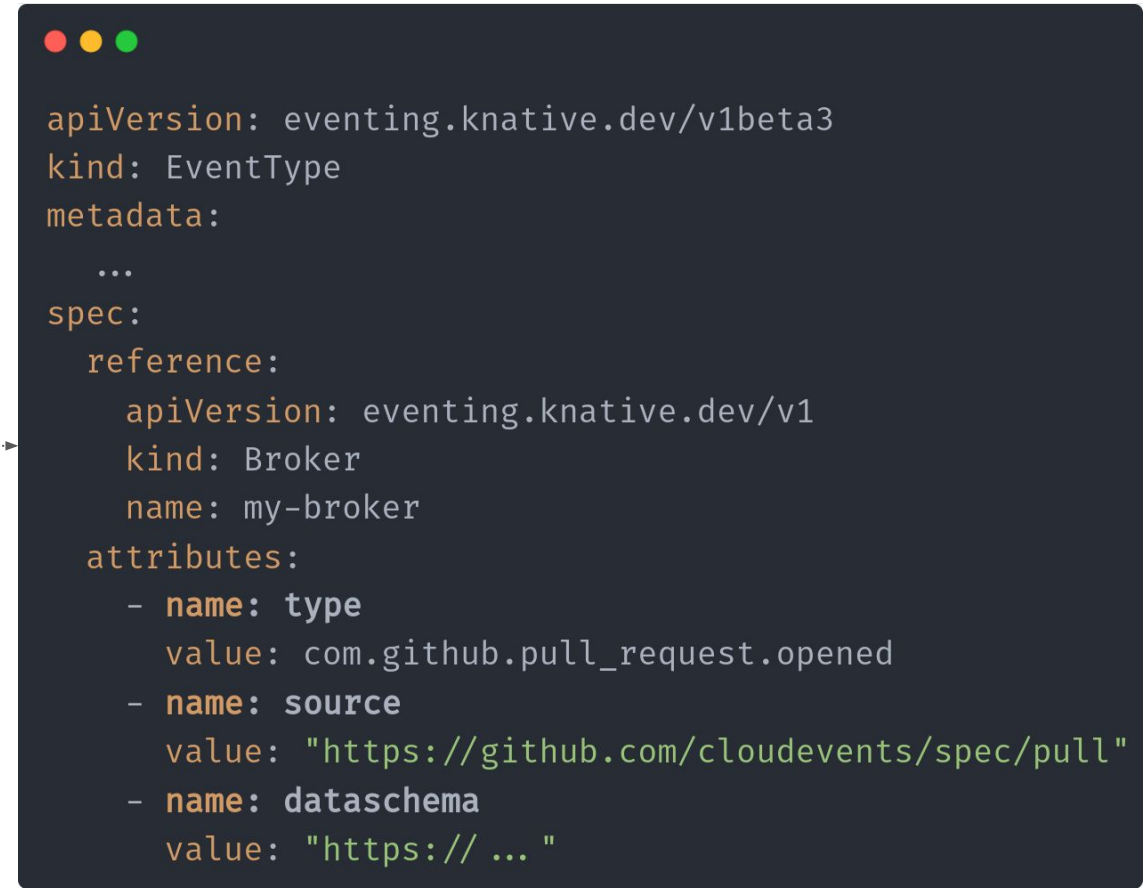
How are EventType catalogs populated?

- Manual Registration
- Automatic Registration (e.g. via Brokers)

Automatic Registration

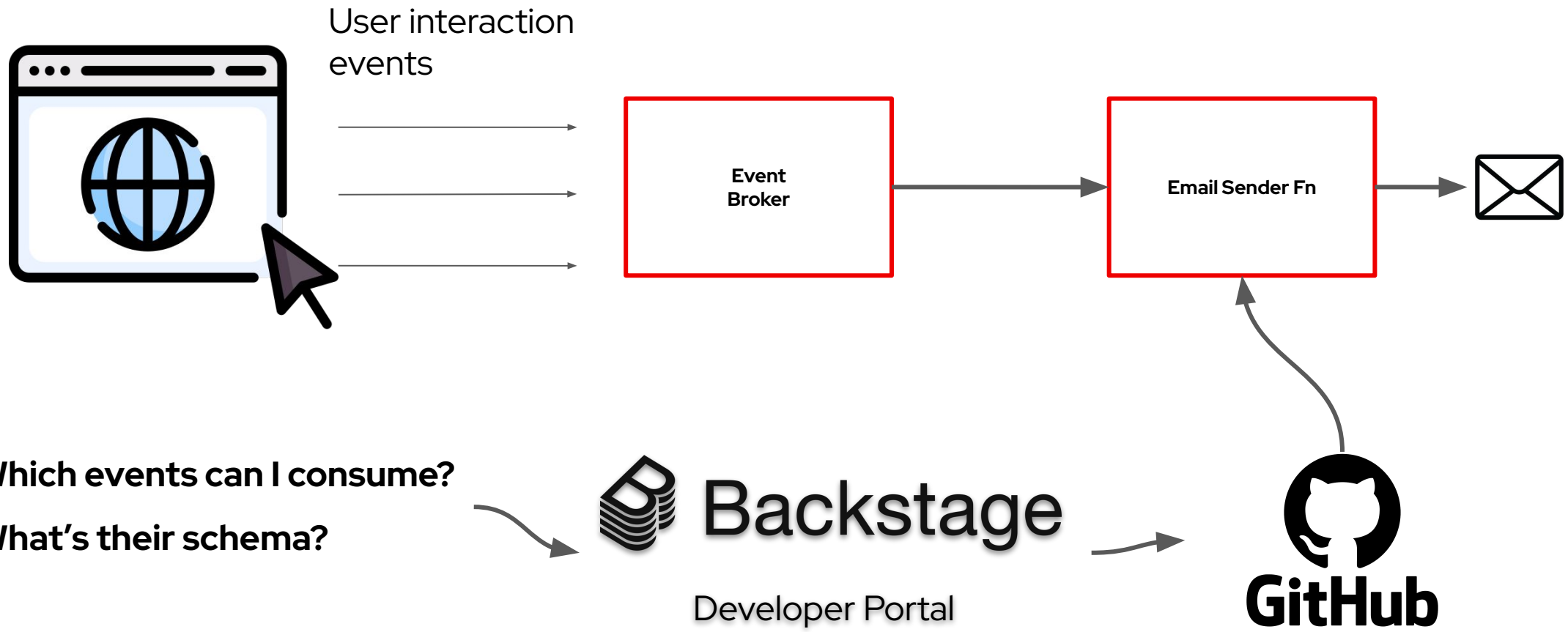
```
{
    "specversion" : "1.0",
    "type" : "com.github.pull_request.opened",
    "source" : "https://github.com/cloudevents/spec/pull",
    "subject" : "123",
    "id" : "A234-1234-1234",
    "time" : "2018-04-05T17:31:00Z",
    "dataschema" : "https:// ... ",
    …
}
```





```
apiVersion: eventing.knative.dev/v1beta3
kind: EventType
metadata:
  ...
spec:
  reference:
    apiVersion: eventing.knative.dev/v1
    kind: Broker
    name: my-broker
  attributes:
    - name: type
      value: com.github.pull_request.opened
    - name: source
      value: "https://github.com/cloudevents/spec/pull"
    - name: dataschema
      value: "https:// ... "
```

Demo


User interaction events
Event Broker
Email Sender Fn
Which events can I consume?
What's their schema?
Backstage
Developer Portal
GitHub

Thank you

Feedback Form