CloudEvents - v1.0 and Beyond: Discovery/Subscriptions

Clemens Vasters - Microsoft
Doug Davis - IBM

Agenda

- CloudEvents Update
- Discovery & Subscription APIs
- Schema Registry

cloudevents

Agenda

- **CloudEvents Update**
- Discovery & Subscriptions APIs
- Schema Registry

CloudEvents

For those who don't know...

cloudevents

CloudEvents

- Defines common metadata for events
- Defines where to find that metadata in the messages

To aid in the delivery of events

No need to understand or parse the business logic just to route

cloudevents

CloudEvents - In Action

HTTP - Binary

```
POST /event HTTP/1.0
Host: example.com
Content-Type: application/json
ce-specversion: 1.0
ce-type: com.bigco.newItem
ce-source: http://bigco.com/repo
ce-id: 610b6dd4-c85d-417b-b58f-3771e532

{
  "action": "newItem",
  "itemID": "93"
}
```

HTTP - Structured

```
POST /event HTTP/1.0
Host: example.com
Content-Type: application/cloudevents+json

{
  "specversion": "1.0",
  "type": "com.bigco.newItem",
  "source": "http://bigco.com/repo",
  "id": "610b6dd4-c85d-417b-b58f-3771e532",
  "datacontenttype": "application/json",
  "data": {
    "action": "newItem",
    "itemID": "93"
  }
}
```

cloudevents

CloudEvents - Summary & Status

Deliverables

- **CloudEvents specification v1.0**
- Transport Bindings (HTTP, AMQP, MQTT, NATS, Kafka)
- Encoding Formats (JSON, AVRO)
- Primer
- SDKs (CSharp, Go, Java, Javascript, PHP, Python, Ruby, Rust)

What's next?

- Customer feedback...
- Additional community pain points...

cloudevents

Agenda

- CloudEvents Update
- **Discovery & Subscription APIs**
- Schema Registry

Delivery of CloudEvents is actually just the end of our story

Before you can deliver events, some negotiations are necessary...

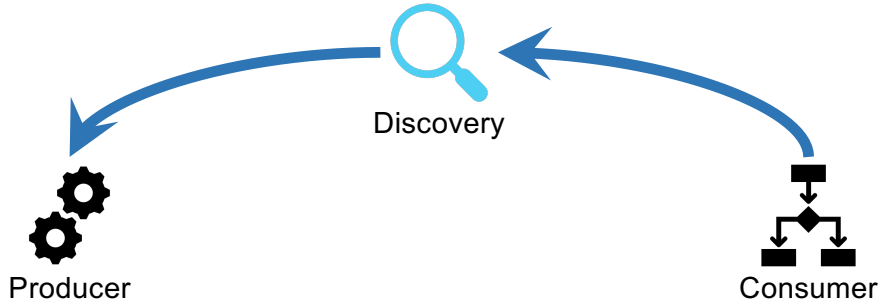
CloudEvents: Discovery

Discovering CloudEvents

- Who produces the events of interest?
- Which events are produced?
- What subscription options are available?
- How to subscribe?

Why?

- Lack of standardization & interop
- Enable automation



Deliverables:

- Discovery Endpoint API Specification
- HTTP / JSON mapping

cloudevents

CloudEvents: Discovery

- A discovery service implements the CloudEvents discovery API
- HTTP and gRPC APIs are well-defined, further protocols may follow

- **Service**
  - Some software entity that emits events
  - Maintains a subscription endpoint
  - The service description enumerates the types of events available

- **Type**
  - A particular kind of event
  - The discovery service allows for a reverse lookup of which services in its scope emit this kind of event

- Discovery services may be federated and share replicated information

cloudevents

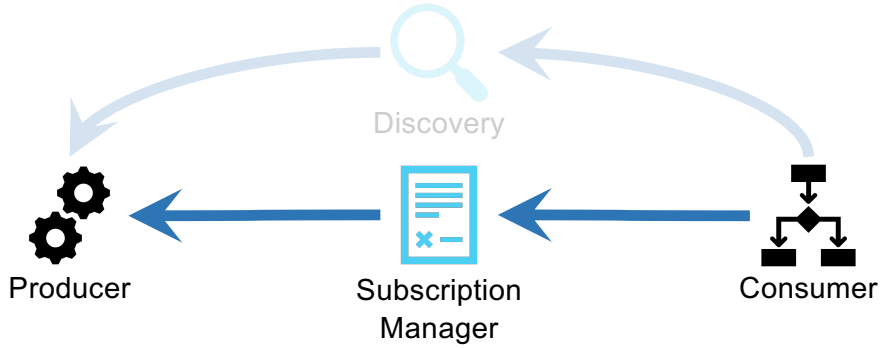
CloudEvents: Subscription

Subscribing to CloudEvents

- Which events to deliver?
- How/where to deliver the events?
  - E.g. Push vs Pull
- Format of the messages?
- How to manage the subscription?

Why?

- Lack of standardization & interop
- Enable automation



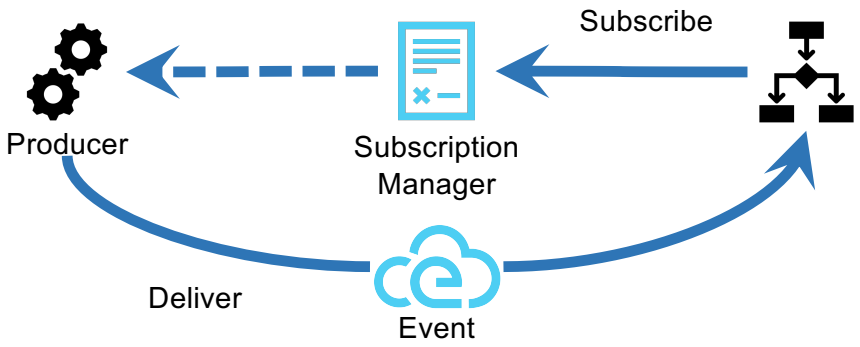
Deliverables:

- Subscriptions API Specification
- HTTP / JSON mapping

cloudevents

CloudEvents: Subscription

- A subscription manager implements the subscription API
- A well-defined HTTP API is defined and bindings to existing subscription mechanisms of other protocols are described
- The subscription manager may be the entity raising events or it may act on its behalf
- Subscriptions may be established for “push” or “pull” delivery style, depending on the delivery protocol



cloudevents

Agenda

- CloudEvents Update
- Discovery & Subscription APIs
- **Schema Registry**

CNCF Schema Registry

- CloudEvents carry payloads with event details, mostly in form of structured data. The same is true for most other messaging and eventing scenarios

- Structured data needs:
  - **Validation:** Check whether received data conforms to a set of structural and syntactic rules. Those rules are expressed in schema documents
  - **Serialization:** in-memory data structures needs to be serialized for network transfer and turned back into in-memory structures after transfer. Efficient serialization encodings rely on external metadata (schemas) to minimize transfer footprints

- The CNCF Schema Registry API, defined in the CloudEvents project, is a vendor-neutral and project-neutral interface to be implemented by schema registries and to be used by validation and serialization clients

cloudevents

CNCF Schema Registry

- Schema Registry core principles:
  - **As simple as possible.** The registry API can be easily implemented on top of a plain file system or cloud blob store and is not overloaded with nonessential complexity
  - **Protocol neutral.** The registry data model is abstractly defined. The HTTP API binding is formally defined with an OpenAPI document, but the registry allows for further protocol bindings. We anticipate having an AMQP binding as well
  - **Scenario neutral.** The registry is explicitly not restricted to use with CloudEvents scenarios. Payload encoding and validation is a universal problem in eventing and messaging. Having separate schema registry infrastructure just for CloudEvents is counterproductive, and there are no competing standards that fit the principles laid out here

cloudevents

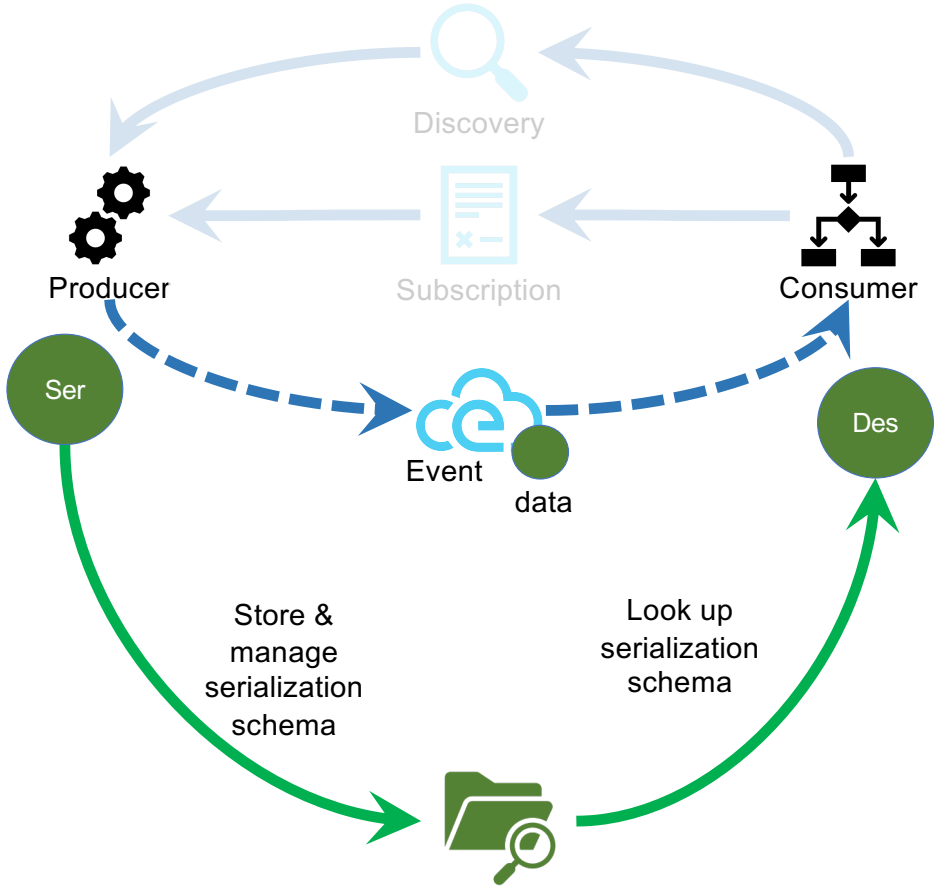
CNCF Schema Registry

Producers:

- Manage serialization and validation schema groups and schemas in the registry
- Use schema to serialize payload data
- Reference the appropriate version of a registered schema in published events/messages

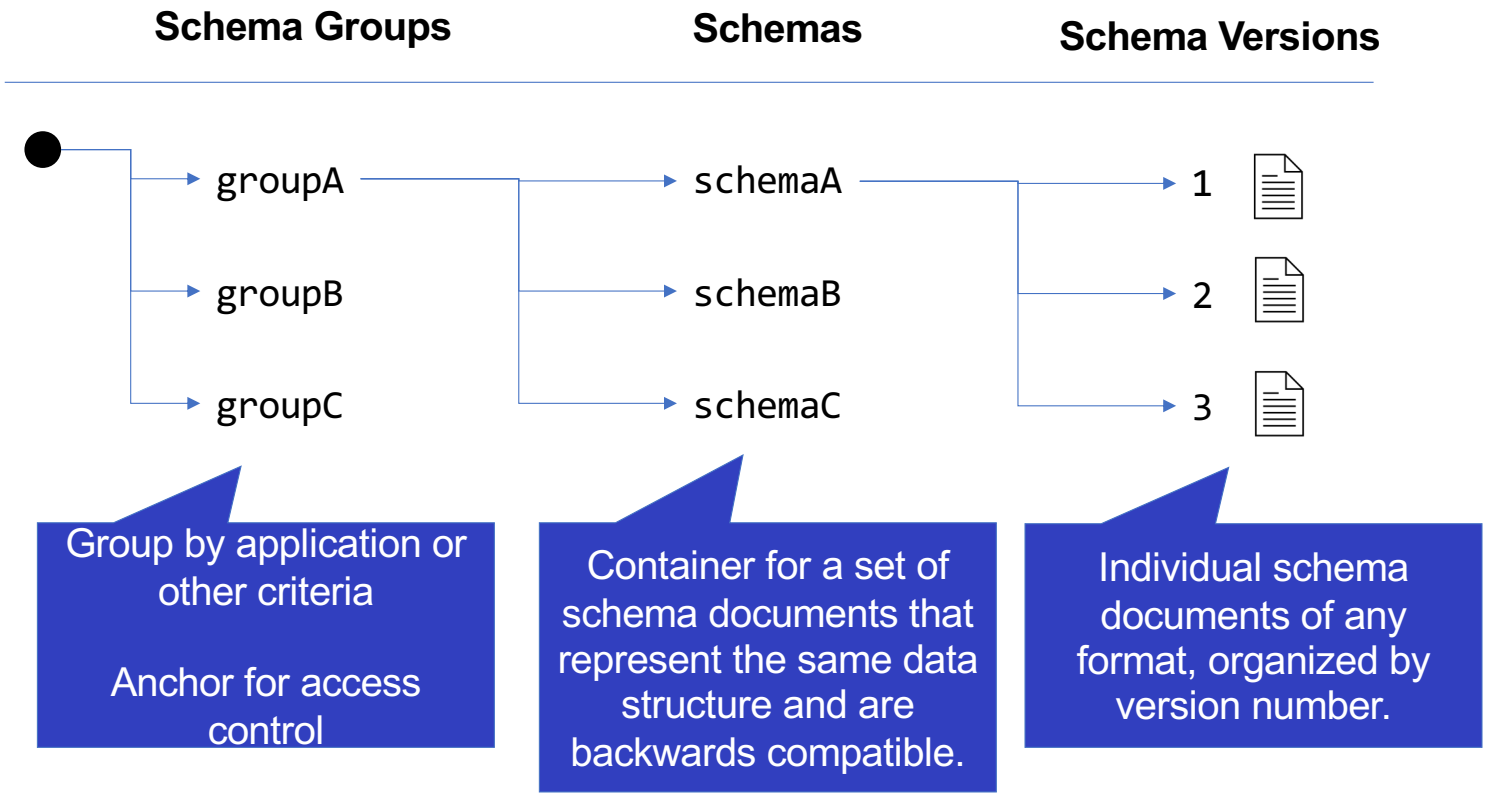
Consumers:

- Look up the validation or serialization schema based on a reference in the message/event metadata
- Validate or deserialize the payload.



cloudevents

CNCF Schema Registry – Structure



cloudevents

Q&A - Thank You!

More information

- CloudEvents: https://cloudevents.io
- Specifications repo: https://github.com/cloudevents/spec
- Weekly calls - Thursdays at 12pm ET (see repo for dial-in info and slack coordinates)

Need more?

- Clemens Vasters - clemensv at microsoft.com | @clemensv
- Doug Davis - dug at us.ibm.com | @duginabox

cloudevents