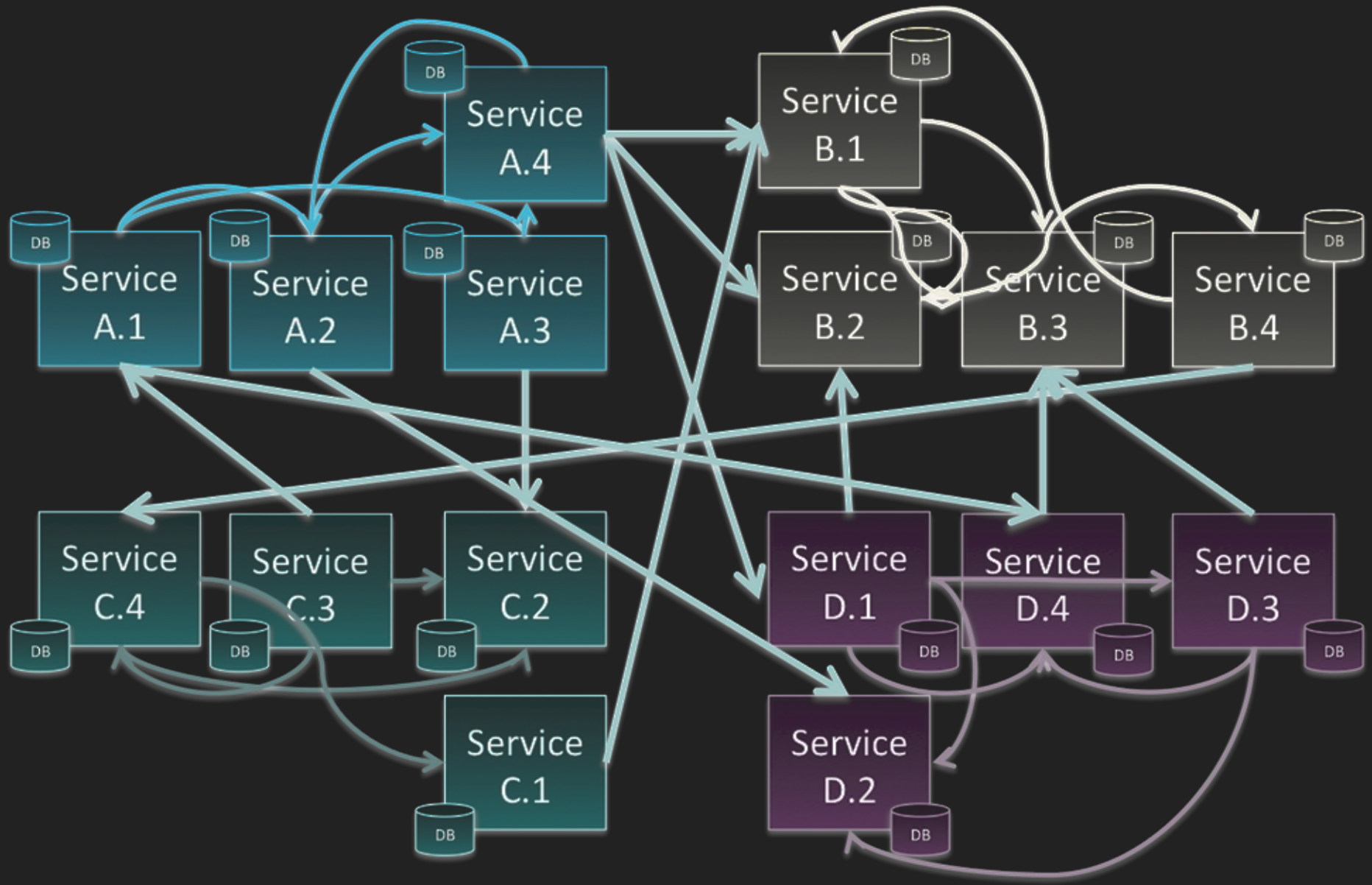
Interfacing with Integrity: A Look at Apicurio Registry

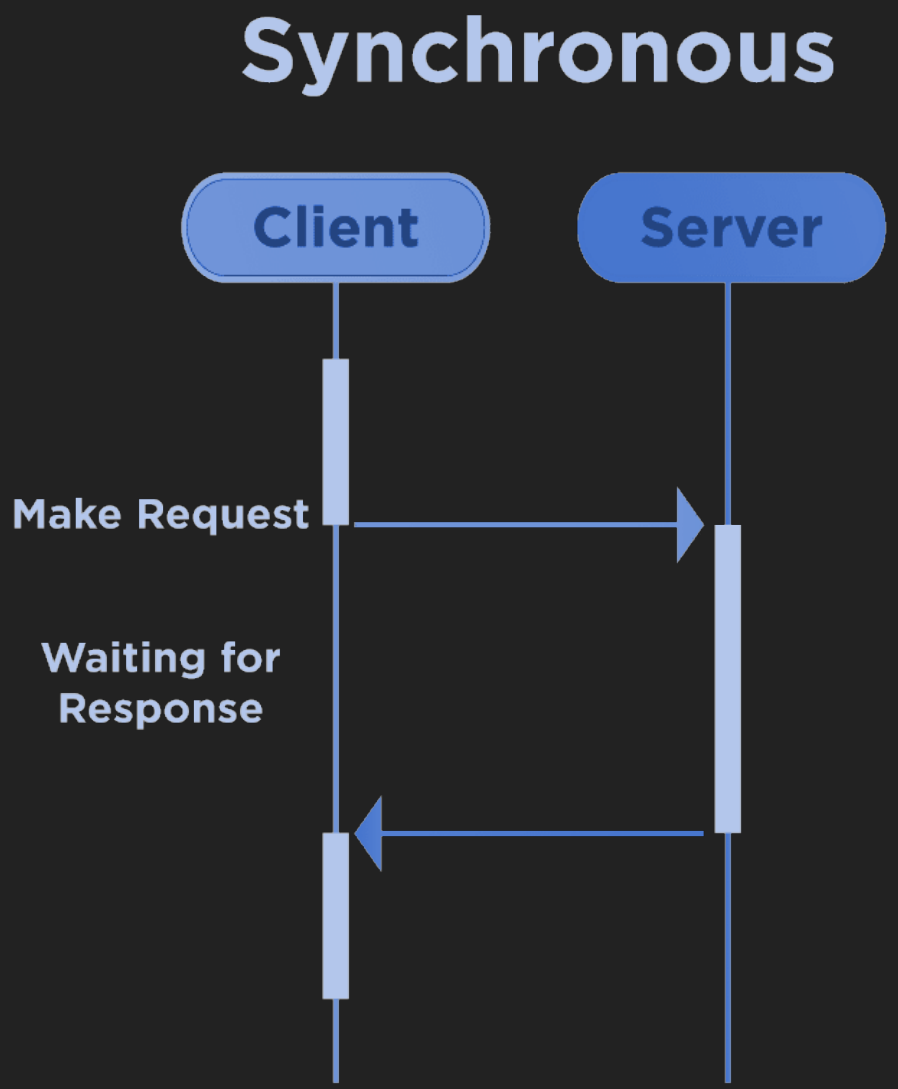
June 2023

Contents

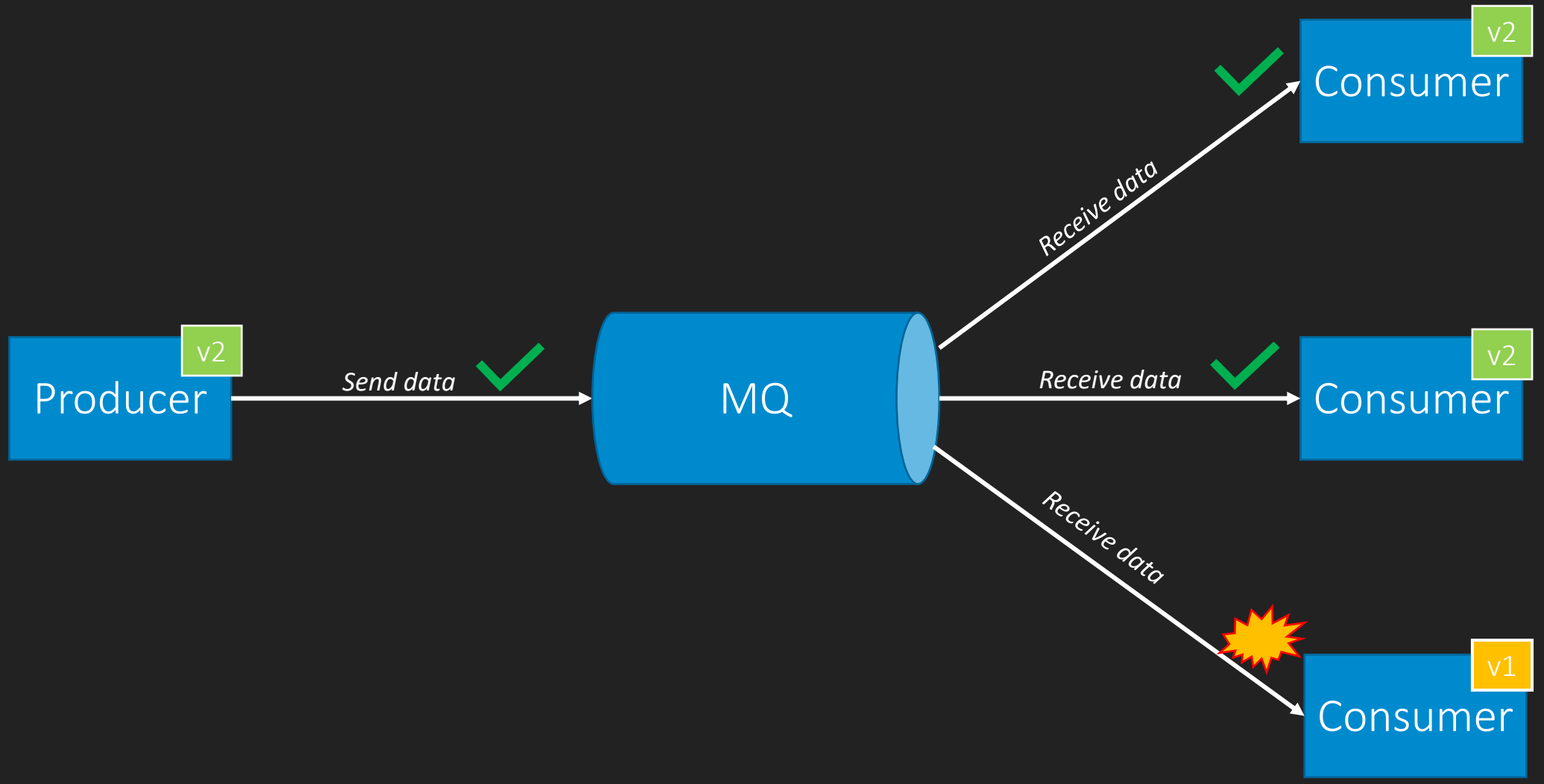
Typical microservice communication way



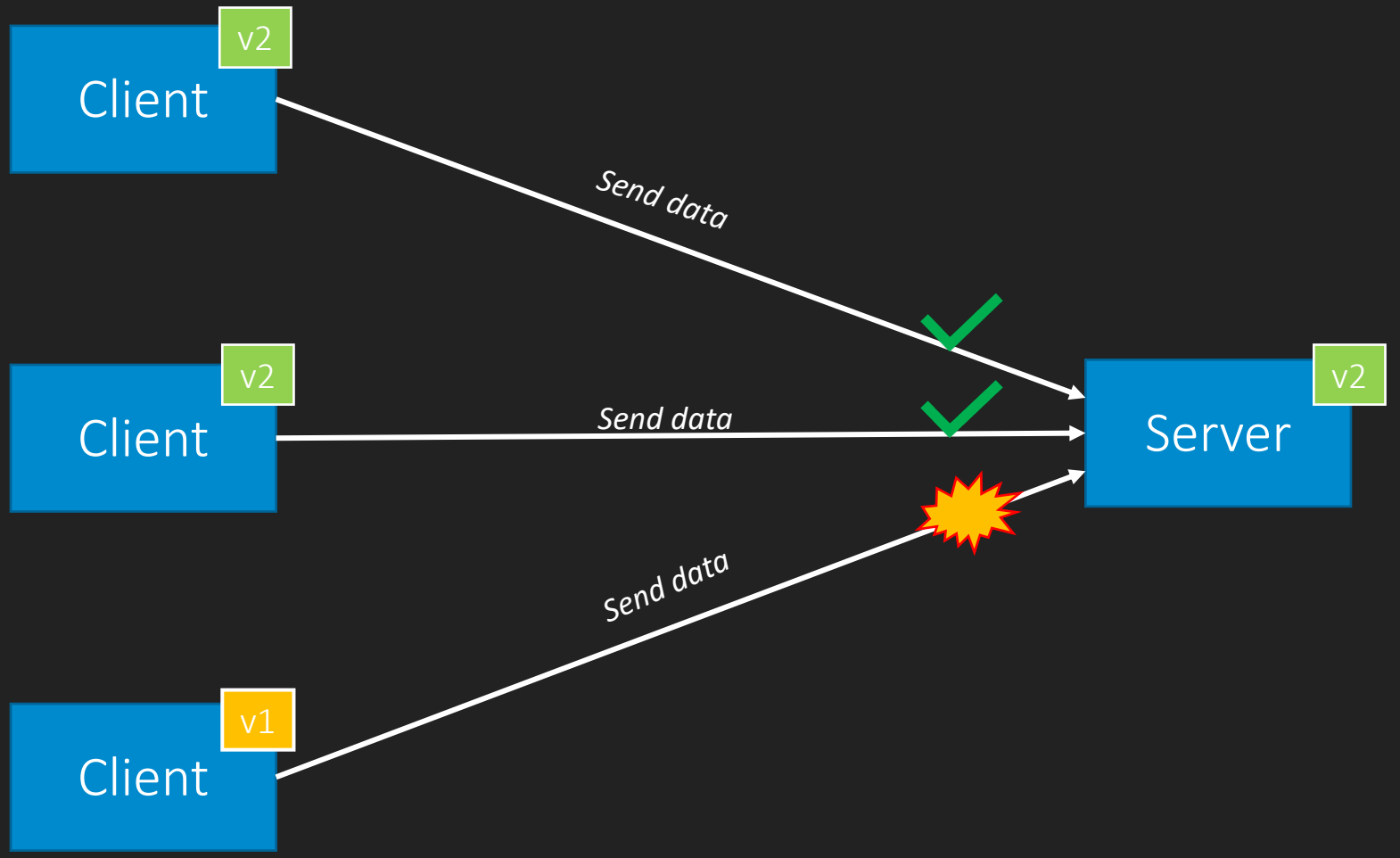
Synchronous vs Asynchronous



Contract-First API Development



Contract-First API Development



TYPICAL ISSUES

Mis-understanding

- Endpoint documentation?
- Data format & validation?
- Infrastructure access?
- Event availability?
- Service providers and consumers compliance?
- Slow time to market?
- Poor quality?

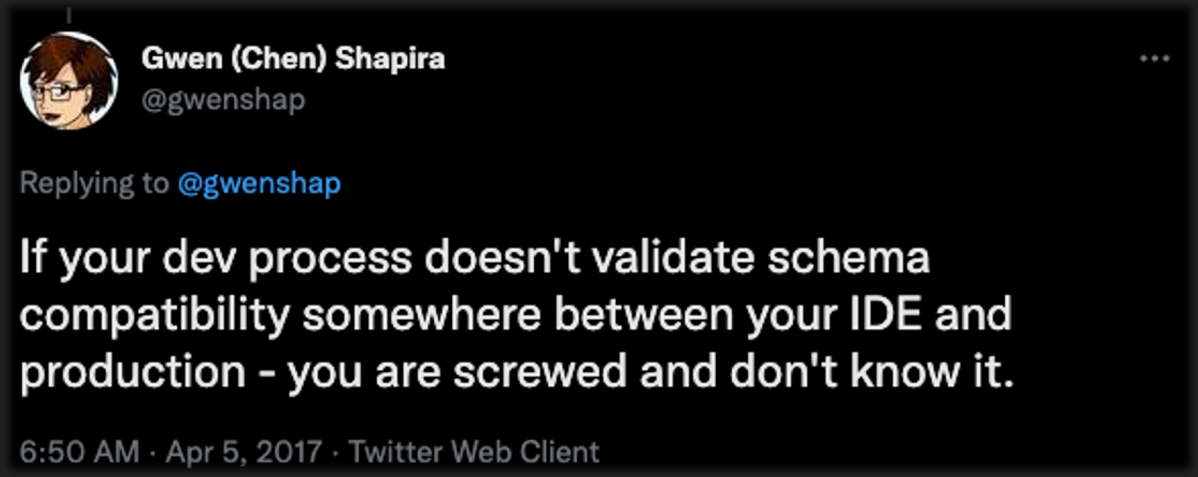
Issues needs to be addressed

- The schemas & API specs are subject to change
- Evolution (and validation) of the schemas & API specs
- Transparency and efficiency
- Central registry where the schemas are stored and accessible
- Prevention of bad data on the consumer side

Fact: a message broker usually is not aware of schema formats

Good practices

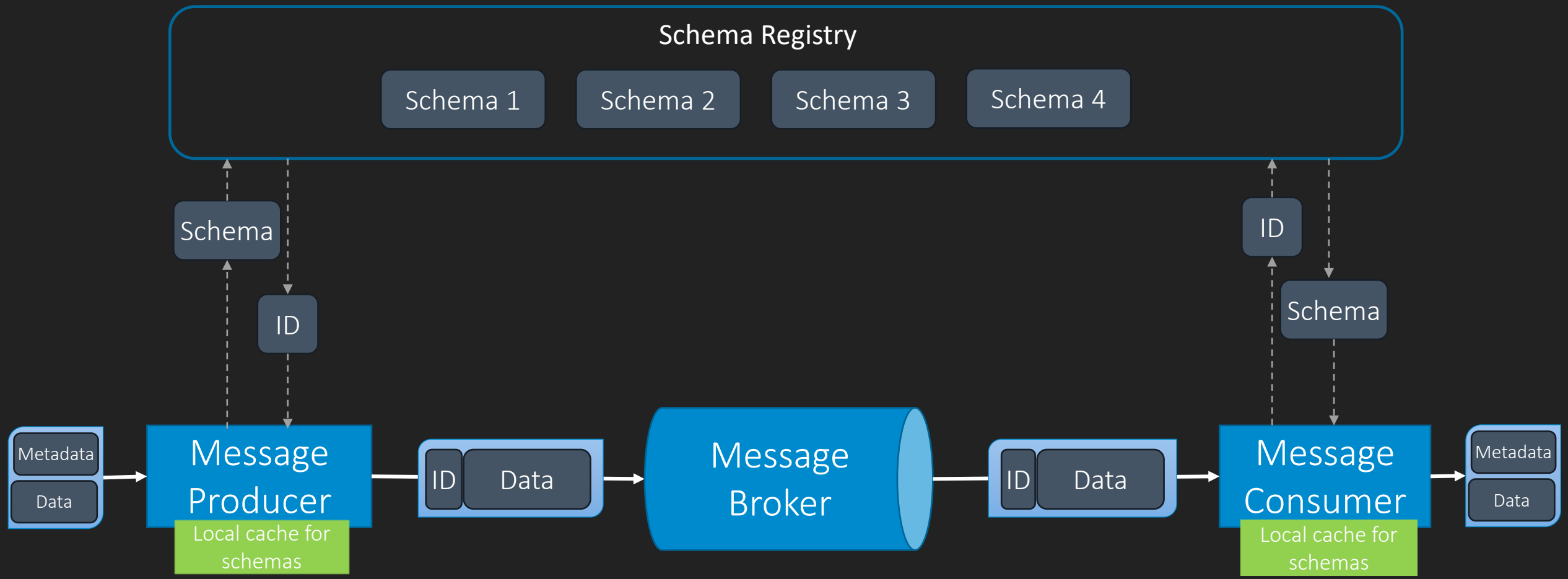
- Add schema (evolution) validation to CI/CD pipelines
- Create a new resource if you need to break compatibility
- Enable schema validation on the consumer side



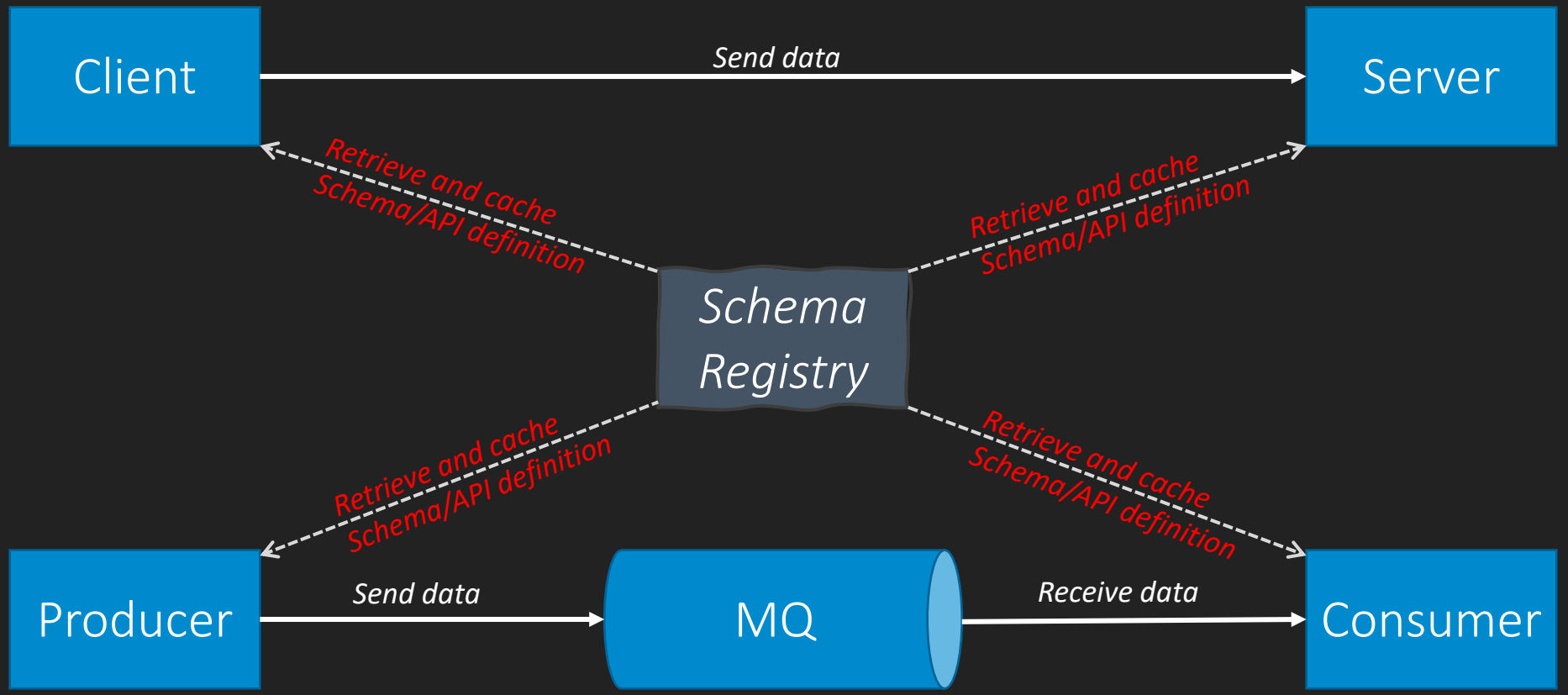
Without Schema Registry


```
{
    "0": {
        "id": "3",
        "title": "Fire",
        "message": "Fire reported at (site) (here) at (tine) - evacuate to parking Lot evacuation points",
        "severityLevel": "1",
        "description": "There is a fire in the mens bathroon next to Chrome nightclub",
        "scheduleAt": "2017-12-12 05:50:00",
        "updatedAt": "2017-12-12 17:50",
        "latitude": 33.8130,
        "longitude": 111.97324
    },
    "1": {
        "id": "4",
        "title": "Power outage",
        "message": "Power outage reported at (site) at (tine) stand by for further information",
        "severityLevel": "2",
        "description": "test blah blah",
        "scheduleAt": "2017-12-12 05:56:00",
        "updatedAt": "2017-12-12 17:56",
        "latitude": 33.281535,
        "longitude": 111.973214
    },
    "2": {
        "id": "5",
        "title": "weather too cold",
        "message": "dont drink it.",
        "severityLevel": "5",
        "description": "the coffee hot and the milk went bad.",
        "scheduleAt": "2017-12-12 06:33:00",
        "updatedAt": "2017-12-12 18:33",
        "latitude": 0,
        "longitude": 0
    }
}
```


With Schema Registry



Using the Schema Registry in the communication

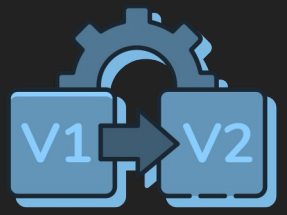


What are the required capabilities of the registry?

Artifact Management

Data Formats

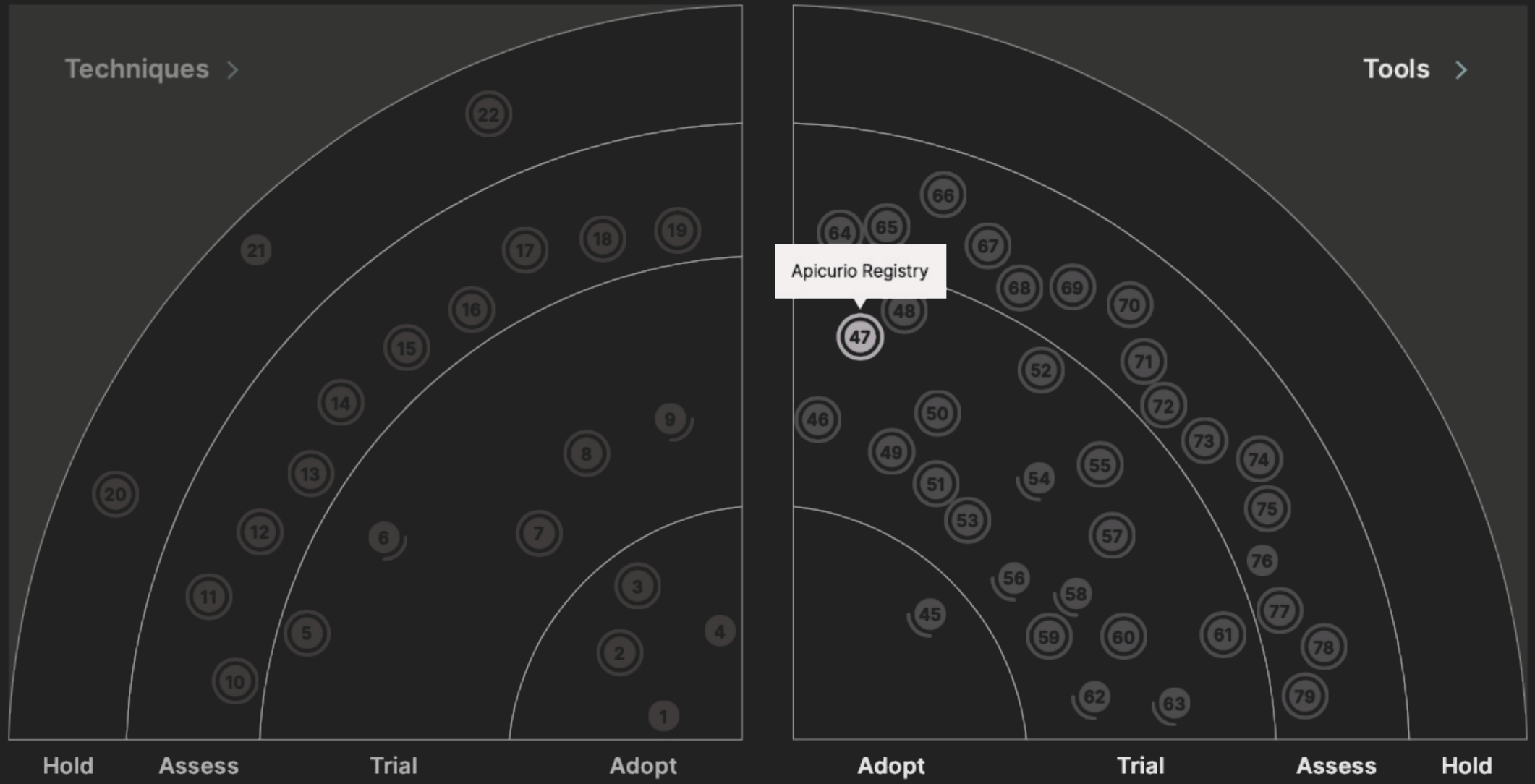
Authentication & Authorization



Evolution & Versioning

Auditability

Thoughtworks Technology Radar Volume 28



Popular OSS Schema Registry Comparison

CONFLUENT

VS

APICURIO

Confluent Schema Registry vs Apicurio Registry

Criteria		Apicurio Registry	Confluent Schema Registry
Schema registration, evolution, and validation		✓	✓
Support schemas	Apache Avro	✓	✓
	JSON Schema	✓	✓
	Protobuf	✓	✓
	GraphQL	✓	✗
	OpenAPI	✓	✗
	AsyncAPI	✓	✗
	WSDL	✓	✗
	XML Schema	✓	✗
Authorization		✓	✗
Auditing (tech. event sourcing)		✓	✗

Confluent Schema Registry vs Apicurio Registry

Criteria		Apicurio Registry	Confluent Schema Registry
Technology stack		Java, Quarkus based	Java, based on Confluent's libraries
Licensing		OSS	OSS with paid addons
Persistence	Kafka	✓	✓
	SQL Database	✓	✗
	In-memory	✓	✗
Authentication	HTTP Basic	✓	✓
	OpenID Connect	✓	✗
Authorization	Role-based	✓	OSS ✗ Paid ✓
	Content-based	✓	✗
GUI		✓	OSS ✗ Paid ✓
Multitenancy		✓	✗

APICURIO

Content rule maturity

- ✓ **Validation Rules**
  - The artifact must have valid content, or the server will reject it
  - Can check for valid syntax (SYNTAX_ONLY)
  - Can also check for valid semantics (FULL)
- ✓ **Compatibility Rules**
  - Determines whether an update is allowed based on configured compatibility requirement setting
  - Multiple compatibility options, including Backwards and Forwards compatible
  - Only relevant for updates (checks the new version against the previous version)
  - Controls the evolution of a single Artifact over time
- ✓ **Integrity Rules**
  - Enforce artifact reference integrity when creating or updating artifacts.
    - All artifact reference integrity checks are enabled (FULL)
    - Detect if there are any duplicate artifact references (NO_DUPLICATES)
    - Detect if there are any references to non-existent artifacts (REFS_EXIST)
    - Ensure that all artifact references are mapped (ALL_REFS_MAPPED)
    - All artifact reference integrity checks are disabled (NONE)

Content rule maturity

Artifact type	Validity rule	Compatibility rule	Integrity rule
Avro	Full	Full	Full
Protobuf	Full	Full	Full
JSON Schema	Full	Full	Mapping detection not supported
OpenAPI	Full	None	Full
WSDL	Full	None	Mapping detection not supported
XSD	Full	None	Mapping detection not supported
AsyncAPI	Syntax Only	None	Full
GraphQL	Syntax Only	None	Mapping detection not supported
Kafka Connect	Syntax Only	None	Mapping detection not supported

Schema Compatibility (for AVRO & JSON Schema)

Compatibility Type	Changes allowed	Check against which schemas	Upgrade first
BACKWARD	• Delete fields • Add optional fields	Last version	Consumers
BACKWARD_TRANSITIVE	• Delete fields • Add optional fields	All previous versions	Consumers
FORWARD	• Add fields • Delete optional fields	Last version	Producers
FORWARD_TRANSITIVE	• Add fields • Delete optional fields	All previous versions	Producers
FULL	• Add optional fields • Delete optional fields	Last version	Any order
FULL_TRANSITIVE	• Add optional fields • Delete optional fields	All previous versions	Any order
NONE	• All changes are accepted	Compatibility checking disabled	Depends

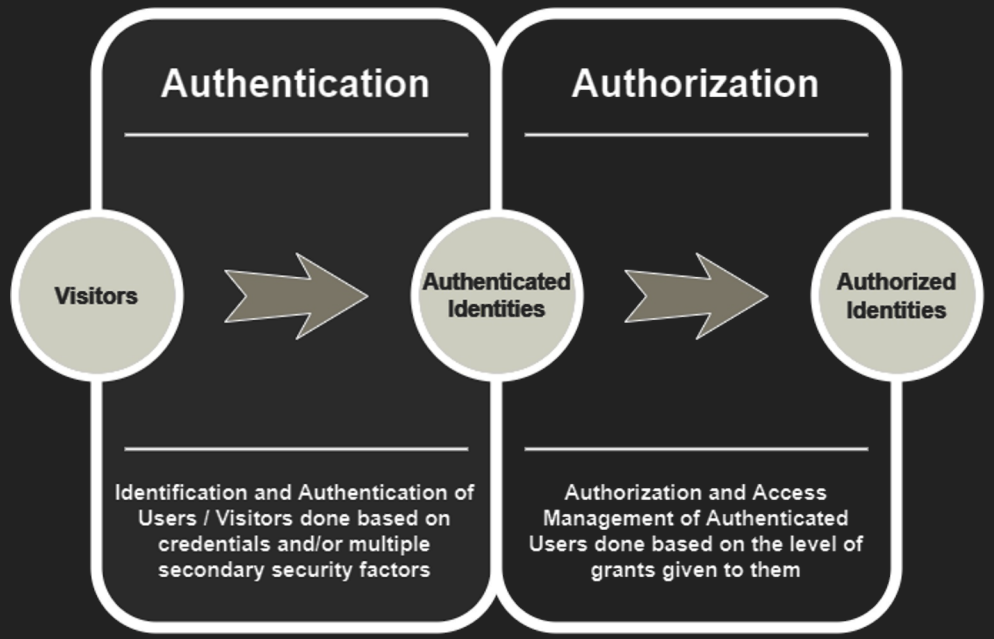
Compatibility tips for AVRO & JSON Schema

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

Authentication & Authorization

Authentication options:

- OpenID Connect with KeyCloak
- HTTP Basic



Authorization options:

- RBAC
- Owner-only

Role	Read artifacts	Write artifacts	Global rules	Summary
ADMIN	Yes	Yes	Yes	Full access to all create, read, update, and delete operations.
DEVELOPER	Yes	Yes	No	Access to create, read, update, and delete operations, except configuring global rules. This role can configure artifact rules.
READ_ONLY	Yes	No	No	Access to read and search operations only. This role cannot configure any rules.

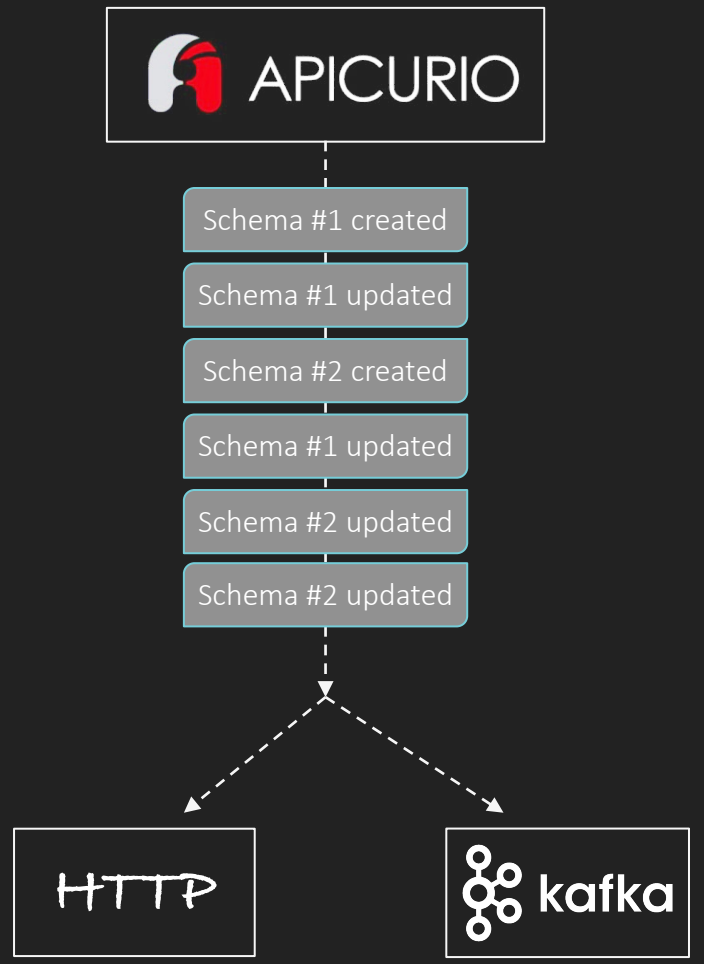
Event sourcing

Apicurio Registry to send events when changes are made to the registry, the event types include:

- io.apicurio.registry.artifact-created
- io.apicurio.registry.artifact-updated
- io.apicurio.registry.artifact-rule-created
- io.apicurio.registry.global-rule-created

The currently implemented protocols:

- HTTP
- Apache Kafka

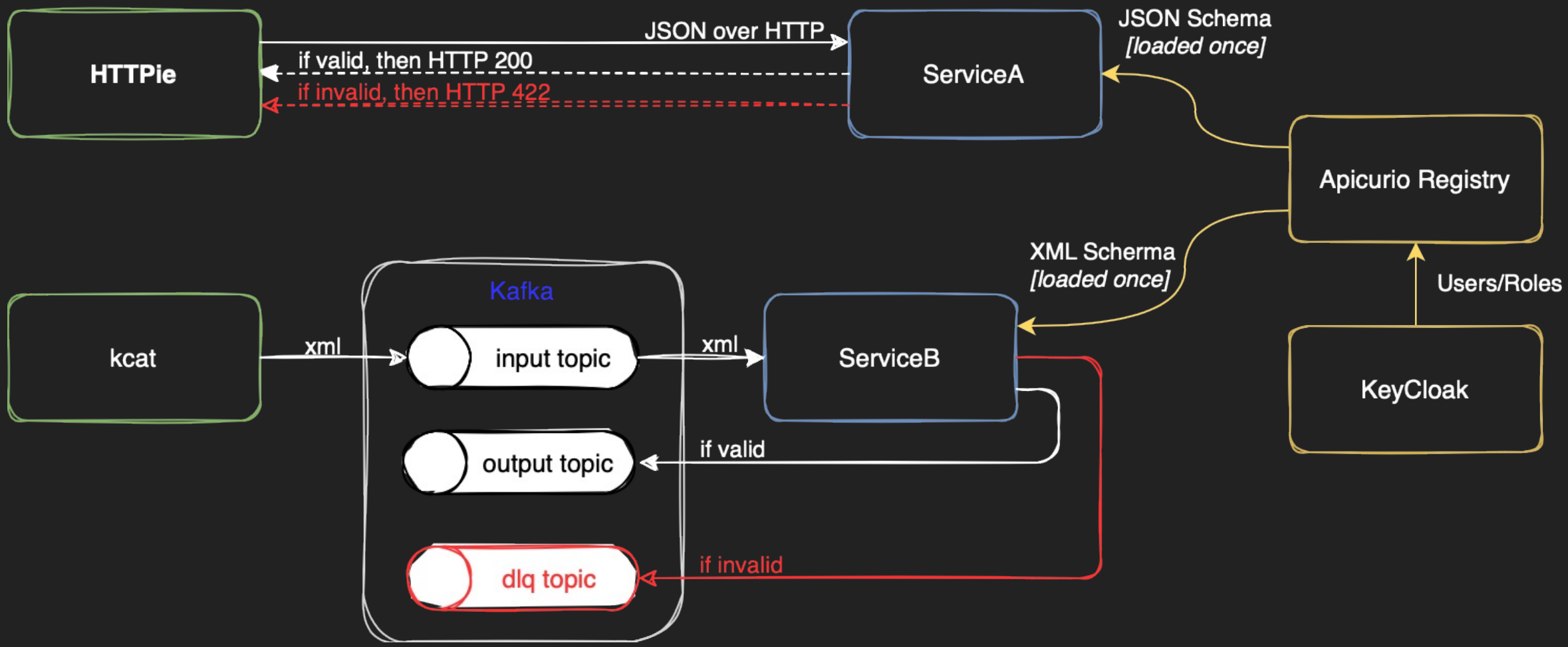


APIs and compartibilities

- /apis/registry/v2 - Core Registry API (Version 2) is the primary API for accessing Apicurio Registry. This is the most recent API and is preferred for all use-cases, *all other APIs are supported for compatibility purposes only*
- /apis/registry/v1 - Core Registry API (Version 1) is the first version of the core Apicurio Registry API, this endpoint is supported to make upgrading consumers of the core registry API easier
- /apis/ccompat/v6 - Apicurio Registry implements the API defined by the *Confluent Schema Registry API (Version 6)*
- /apis/ccompat/v7 - Apicurio Registry implements the API defined by the *Confluent Schema Registry API (Version 7)*
- /apis/cncf/v0 - The CNCF community has worked to create an open standard for schema registries, for cloud events and for other use-cases i.e. *CNCF Schema Registry API (Version 0)*

DEMO

Demo



The sources are available by the link: https://github.com/stn1slv/meetup-schema-registry


ANY
QUESTIONS?