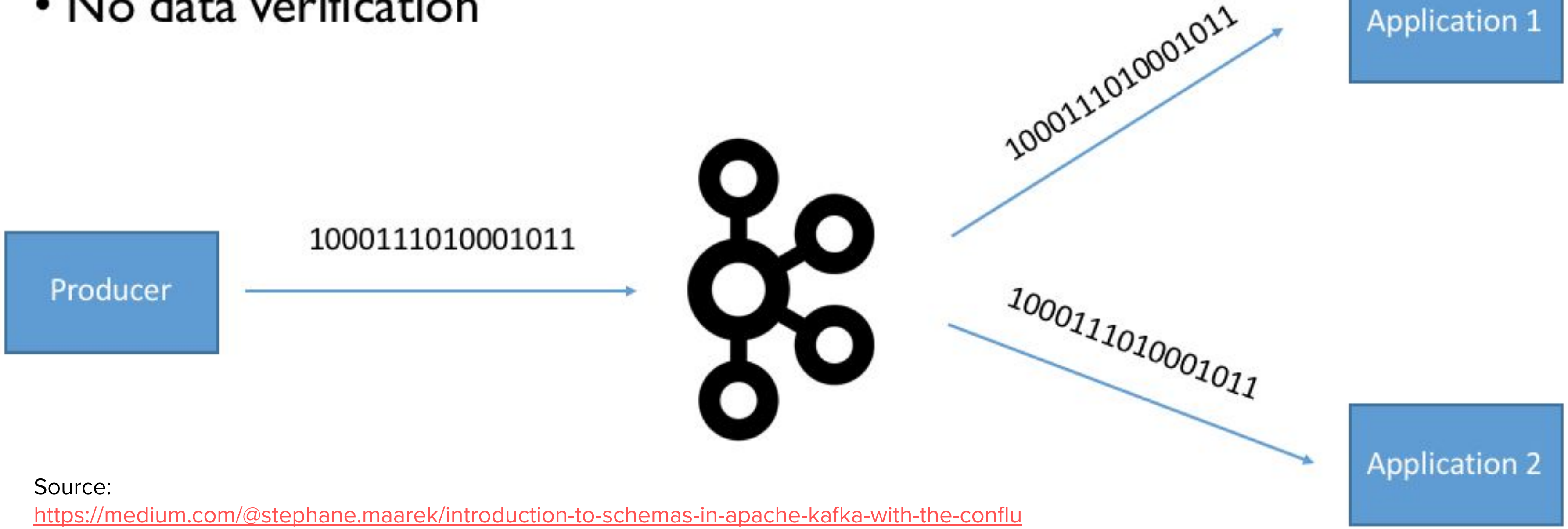
Schema Registry

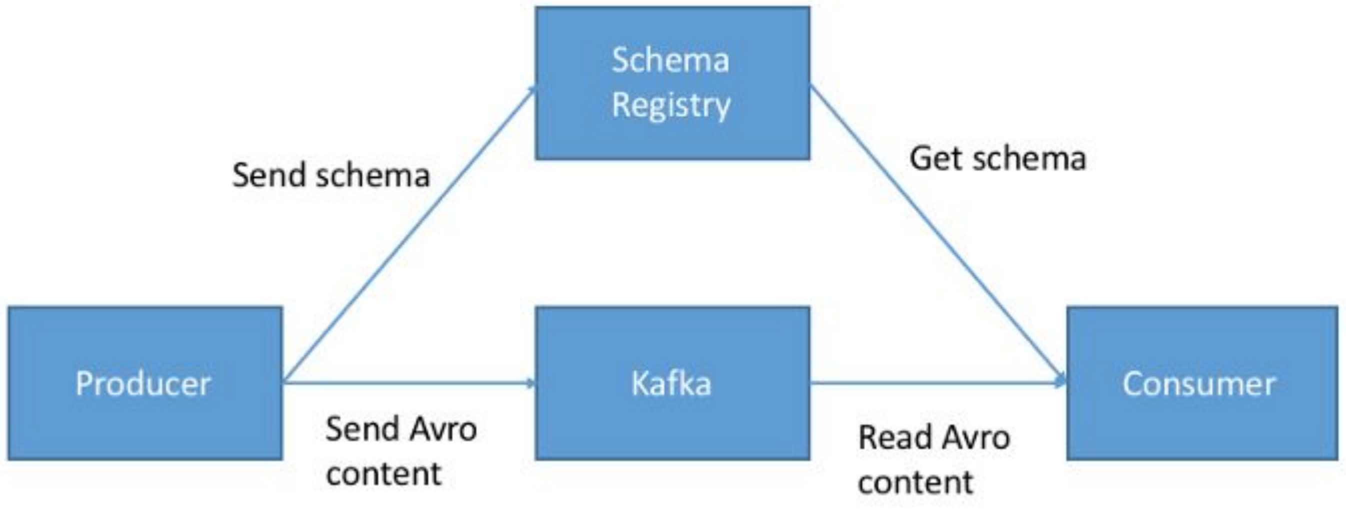
Apache Kafka

- Kafka takes bytes as an input and publishes them
- No data verification



Source:
https://medium.com/@stephane.maarek/introduction-to-schemas-in-apache-kafka-with-the-confluent-schema-registry-3bf55e401321

Schema Registry



Source:
https://medium.com/@stephane.maarek/introduction-to-schemas-in-apache-kafka-with-the-confluent-schema-registry-3bf55e401321

Kafka Schema Registry

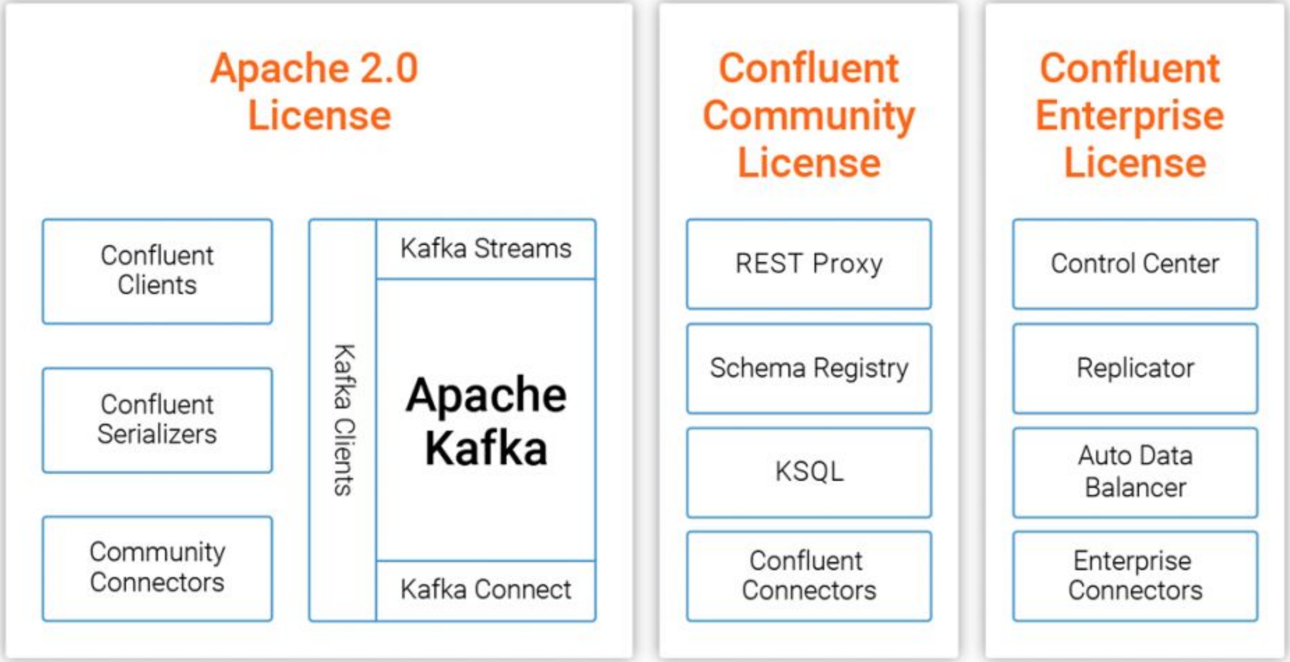
Kafka Schema Registry provides serializers that plug into Apache Kafka clients that handle message schema storage and retrieval for Kafka messages that are sent in the **Avro,Protobuf and JSON forma**t. It used to be an OSS project by Confluent, but is now under the Confluent community license.

The Schema Registry additionally serves the below purposes:

Stores and retrieves schemas for producers and consumers.

Enforce backward/forward /full compatibility on Topics.

Commercial licensing



Source: https://www.confluent.io/blog/license-changes-confluent-platform/

Schema Registry

When using the Confluent Schema Registry, producers don’t have to send schema — just the schema ID, which is unique. The consumer uses the schema ID to look up the full schema from the Confluent Schema Registry if it's not already cached. Since you don’t have to send the schema with each set of records, this saves time. Not sending the schema with each record or batch of records speeds up the serialization, as only the ID of the schema is sent.

Schema compatibility

When you start modifying schemas you need to take into account a number of issues: whether to upgrade consumers or producers first; how consumers can handle the old events that are still stored in Kafka; how long we need to wait before we upgrade consumers; and how old consumers handle events written by new producers => Schema compatibility

Schema compatibility checking is implemented in Schema Registry by versioning every single schema. The **compatibility type** determines how Schema Registry compares the new schema with previous versions of a schema, for a given subject . When a schema is first created for a subject, it gets a unique id and it gets a version number, i.e., version 1. When the schema is updated (if it passes compatibility checks), it gets a new unique id and it gets an incremented version number, i.e., version 2.

Note: A subject refers to the name under which the schema is registered. If you are using Schema Registry for Kafka, then a subject refers to either a “<topic>-key” or “<topic>-value” depending on whether you are registering the key schema for that topic or the value schema. E.g. sales-avro-value

Schema Compatibility

The schema compatibility checks can be configured globally or per subject.

The compatibility value will be one of the following:

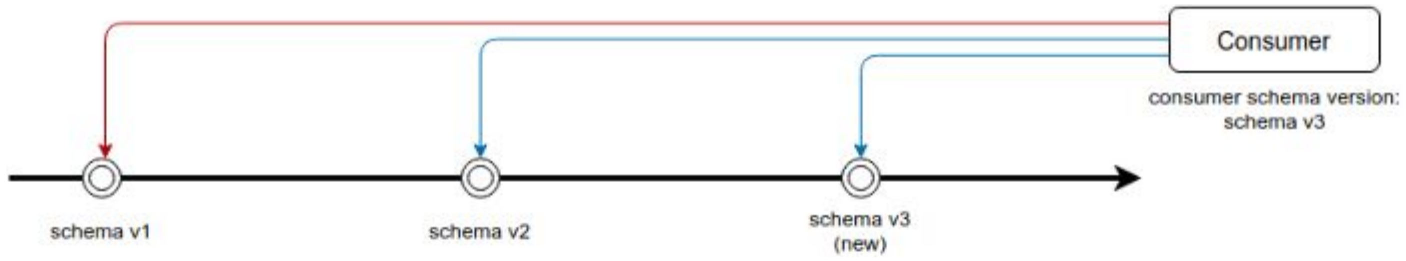
None: Don’t check for schema compatibility.

Forward: Check to make sure the last schema version is forward-compatible with new schemas.

Backward (default): Make sure the new schema is backward-compatible with the latest.

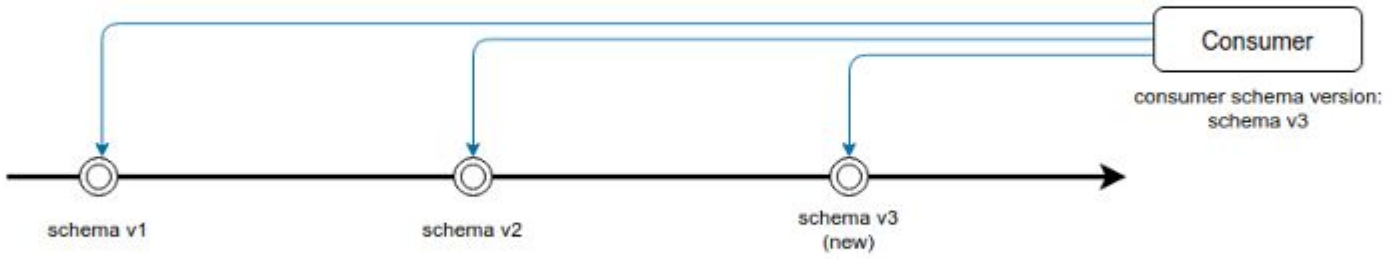
Full: Make sure the new schema is forward- and backward-compatible from the latest to newest and from the newest to latest.

Schema compatibility - backward compatibility



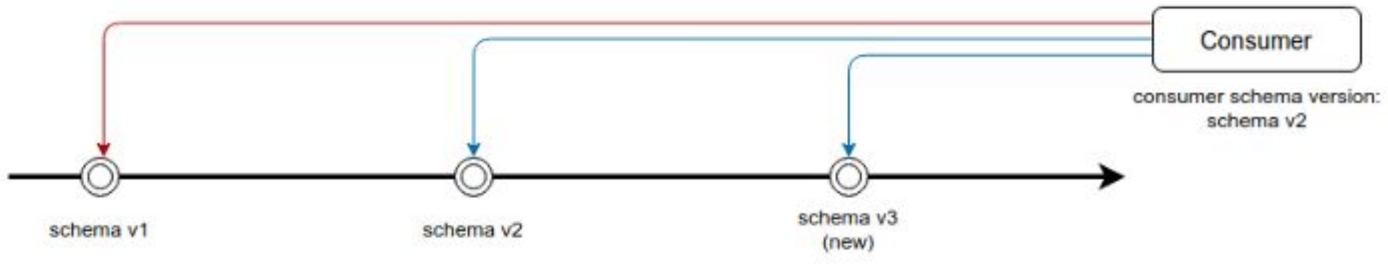
The BACKWARD compatibility means that consumers using the new schema can read data produced with the last schema version but it does not assure compatibility with the versions before the last version.
A schema is backwards compatible as long as only the following operations are executed will evolving it
* **delete fields**
* **add optional fields**
* **add mandatory fields with a meaningful default value**
* **convert a mandatory field into optional**

Schema compatibility - Backward_transitive compatibility mode



The BACKWARD_TRANSITIVE compatibility means that consumers using the new schema can read data produced by all previous schema versions.

Schema compatibility - Forward compatibility mode



The FORWARD compatibility means that consumers using the last schema can read data produced with the new schema version but the new schema version does not assure compatibility with the versions before the last version.

*** add fields (optional or mandatory, with or without a default value)**
*** remove optional fields**
*** remove mandatory fields associated with a meaningful default value**
*** convert an optional field into mandatory**