**Apache Avro and Kafka - Helpful Documentation**

* [**confluent\_kafka\_python Avro Producer**](https://docs.confluent.io/current/clients/confluent-kafka-python/index.html?highlight=partition#confluent_kafka.avro.AvroProducer)
* [**confluent\_kafka\_python Avro Consumer**](https://docs.confluent.io/current/clients/confluent-kafka-python/index.html?highlight=partition#confluent_kafka.avro.AvroConsumer)

**Schema Registry - Summary**

* Provides an HTTP REST API for managing Avro schemas
* Many Kafka clients natively support Schema Registry interactions for you
* Reduces network overhead, allowing producers and consumers to register schemas one time
* Simplifies using Avro, reducing the barrier to entry for developers
* Uses a Kafka topic to store state
* Deployed as one or more web servers, with one leader
* Uses ZooKeeper to manage elections

**Schema Registry - Optional Further Research**

* [**confluent\_kafka\_python Avro and Schema Registry support**](https://docs.confluent.io/current/clients/confluent-kafka-python/index.html?highlight=partition#module-confluent_kafka.avro)
* [**Schema Registry Overview**](https://docs.confluent.io/current/schema-registry/index.html)
* [**Schema Registry HTTP API Documentation**](https://docs.confluent.io/current/schema-registry/develop/api.html)

## Schema Compatibility

* The process of schema change is known as Schema Evolution
* Schema Evolution is caused by a modification to an existing data schema
  + Adding or removing a field
  + Making a field optional
  + Changing a field type
* Schema Registry can track schema compatibility between schemas
  + Compatibility is used to determine whether or not a particular schema version is usable by a data consumer
  + Consumers may opt to use this compatibility information to preemptively refuse to process data that is incompatible with its current configuration
  + Schema Registry supports [**four categories of compatibility**](https://docs.confluent.io/current/schema-registry/avro.html)
  + Backward / Backward Transitive
  + Forward / Forward Transitive
  + Full / Full Transitive
  + None
* Managing compatibility requires both producer and consumer code to determine the compatibility of schema changes and send those updates to Schema Registry

**Backward Compatibility - Key Points**

* [**Backward compatibility**](https://docs.confluent.io/current/schema-registry/avro.html#backward-compatibility) means that consumer code developed against the most recent version of an Avro Schema can use data using the prior version of a schema without modification.
  + The deletion of a field or the addition of a new optional field are backward compatible changes.
  + Update consumers before updating producers to ensure that consumers can handle the new data type
* The BACKWARD compatibility type indicates compatibility with the current version (N) and the immediately prior version (N-1)
  + Unless you specify otherwise, Schema Registry always assumes that changes are BACKWARD compatible
* The BACKWARD\_TRANSITIVE compatibility type indicates compatibility with *all* prior versions (1 → N)

## Forward Compatibility

* [**Forward compatibility**](https://docs.confluent.io/current/schema-registry/avro.html#forward-compatibility) means that consumer code developed against the previous version of an Avro Schema can consume data using the newest version of a schema without modification
* The deletion of an optional field or the addition of a new field are forward compatible changes
* Producers need to be updated before consumers
* The FORWARD compatibility type indicates that data produced with the latest schema (N) is usable by consumers using the previous schema version (N-1)
* The BACKWARD\_TRANSITIVE compatibility type indicates that data produced with the latest schema (N) is usable by all consumers using any previous schema version (1 → N-1)

**Lesson Recap**

In this lesson, you learned what a data schema is, and how they can help us better deal with change as it occurs within our data streams. Specifically we learned how Apache Avro works and how it integrates with tools like Schema Registry to provide producers and consumers information about the data they are consuming. Finally, we learned how to decide if a change is backward, forward, or full compatible.

**Glossary of Key Terms in Lesson (same as presented at beginning of lesson)**

* Data Schema - Define the shape of a particular kind of data. Specifically, data schemas define the expected fields, their names, and value types for those fields. Data schemas may also indicate whether fields are required or optional.
* Apache Avro - A data serialization framework which includes facilities for defining and communicating data schemas. Avro is widely used in the Kafka ecosystem and data engineering generally.
* Record (Avro) - A single encoded record in the defined Avro format
* Primitive Type (Avro) - In Avro, a primitive type is a type which requires no additional specification - null, boolean, int, long, float, double, bytes, string.
* Complex Type (Avro) - In Avro, a complex type models data structures which may involve nesting or other advanced functionality: records, enums, maps, arrays, unions, fixed.
* Schema Evolution - The process of modifying an existing schema with new, deleted, or modified fields.
* Schema Compatibility - Determines whether or not two given versions of a schema are usable by a given client
* Backward Compatibility - means that consumer code developed against the most recent version of an Avro Schema can use data using the prior version of a schema without modification.
* Forward Compatibility - means that consumer code developed against the previous version of an Avro Schema can consume data using the newest version of a schema without modification.
* Full Compatibility - means that consumers developed against the latest schema can consume data using the previous schema, and that consumers developed against the previous schema can consume data from the latest schema as well. In other words, full compatibility means that a schema change is both forward and backward compatible.
* None Compatibility - disables compatibility checking by Schema Registry.