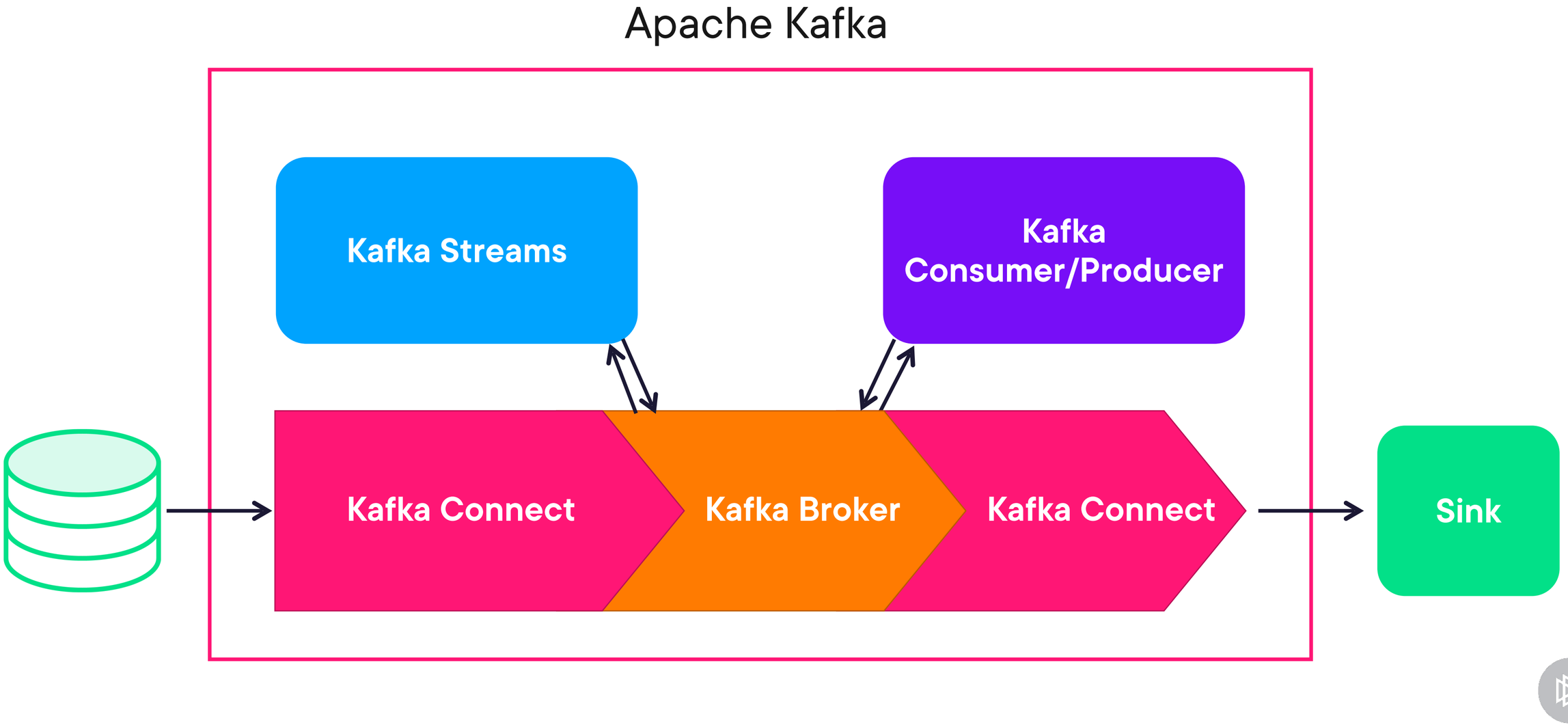
Kafka Connect

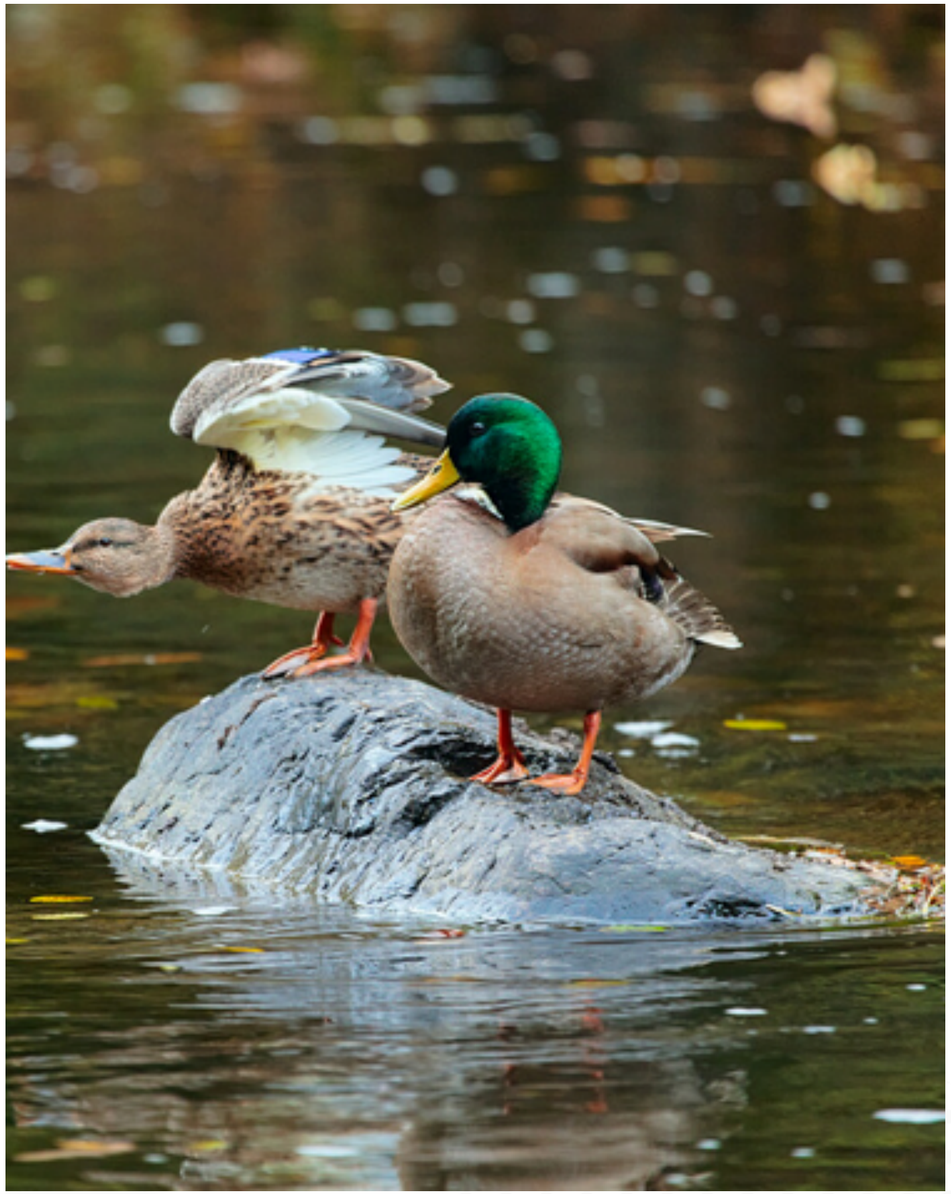
Axel Sirota

AI and Cloud Consultant

@AxelSirota

ETL with Apache Kafka

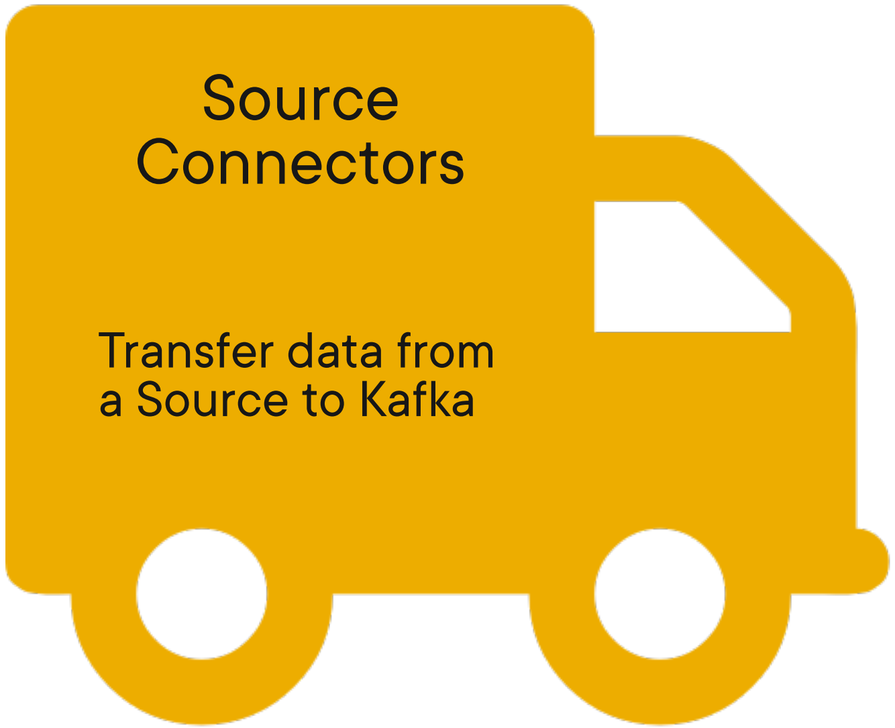




One important statement is that Kafka Connect is not an ETL solution per se, it only connects.

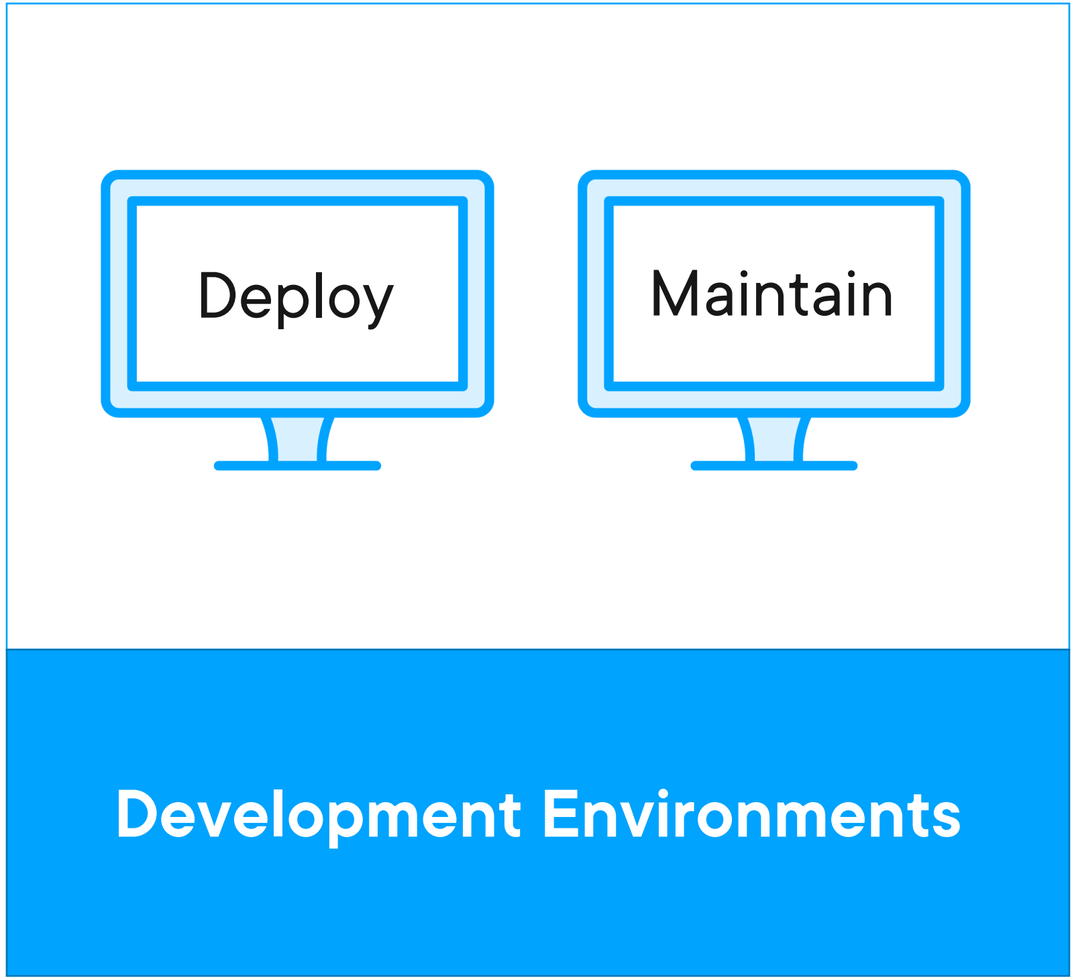
But with the help of the correct plugins it can have some ETL capabilities

Connectors

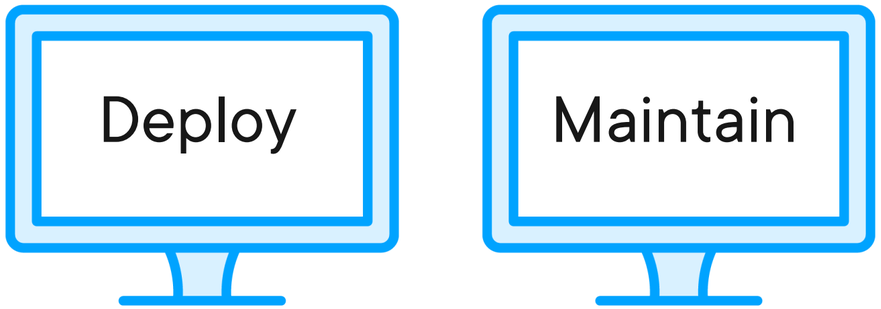


https://www.confluent.io/product/connectors/

Standalone

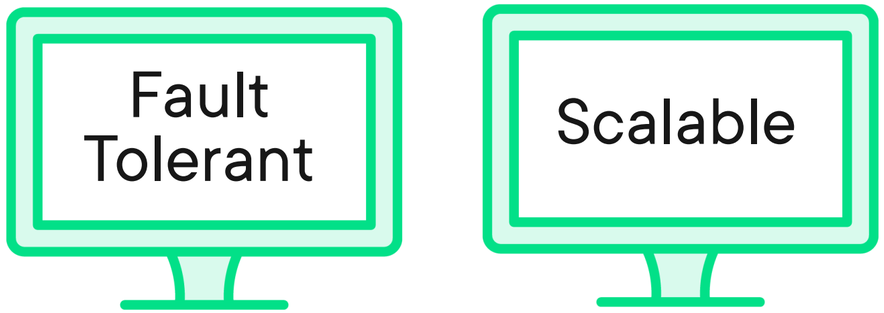


Standalone



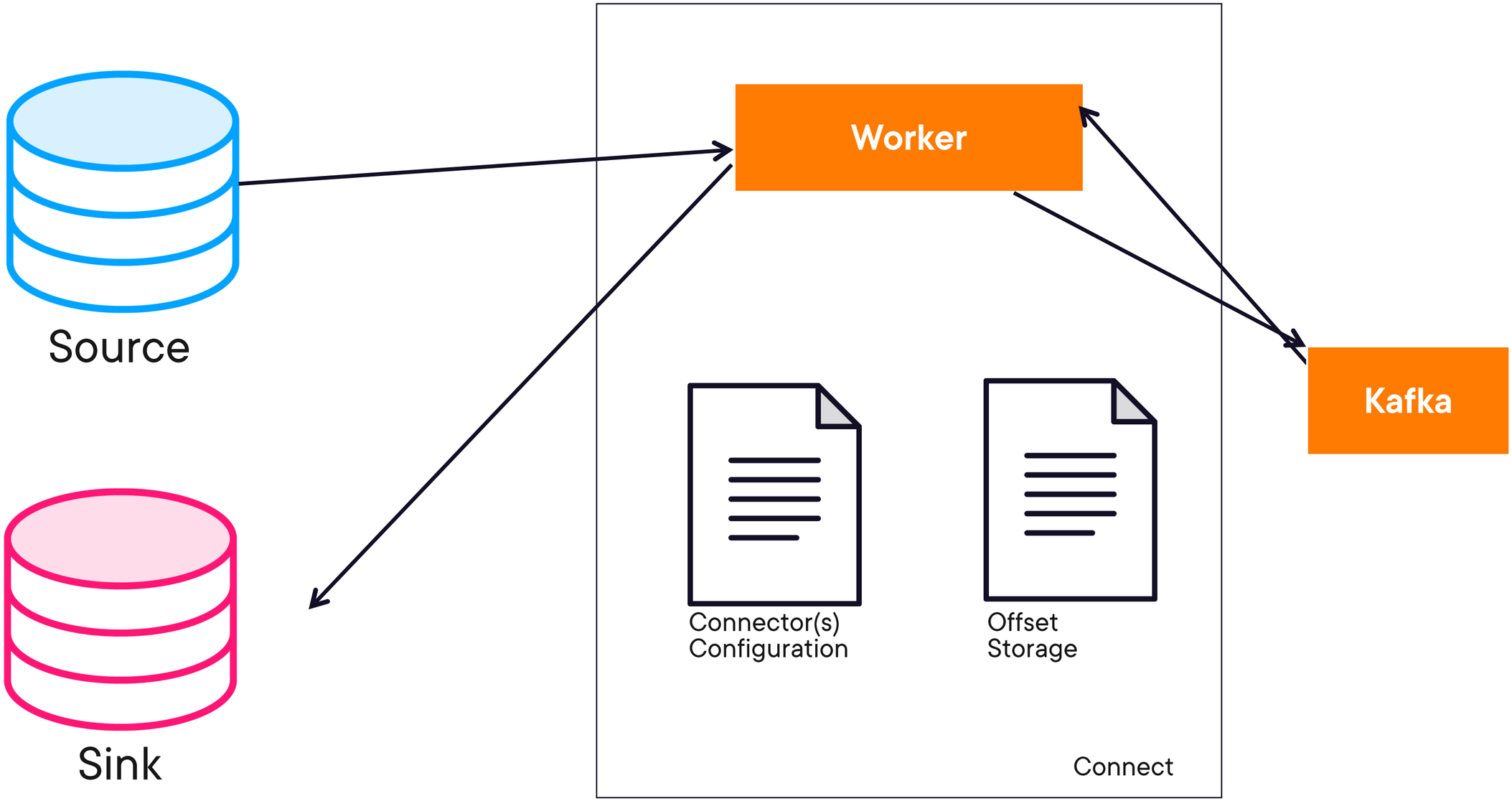
Development Environments

Distributer (Cluster)

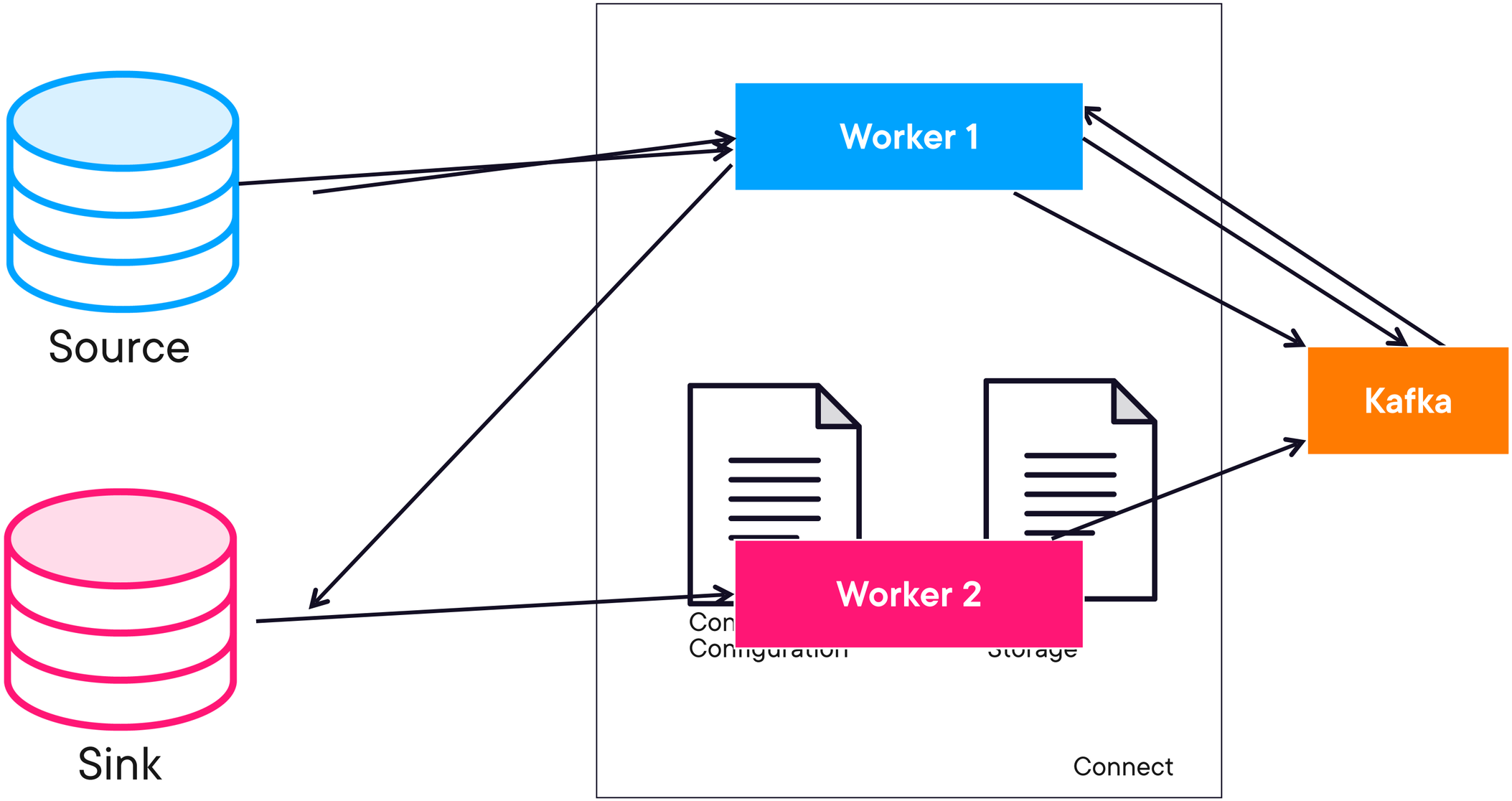


Production (Like) Environments

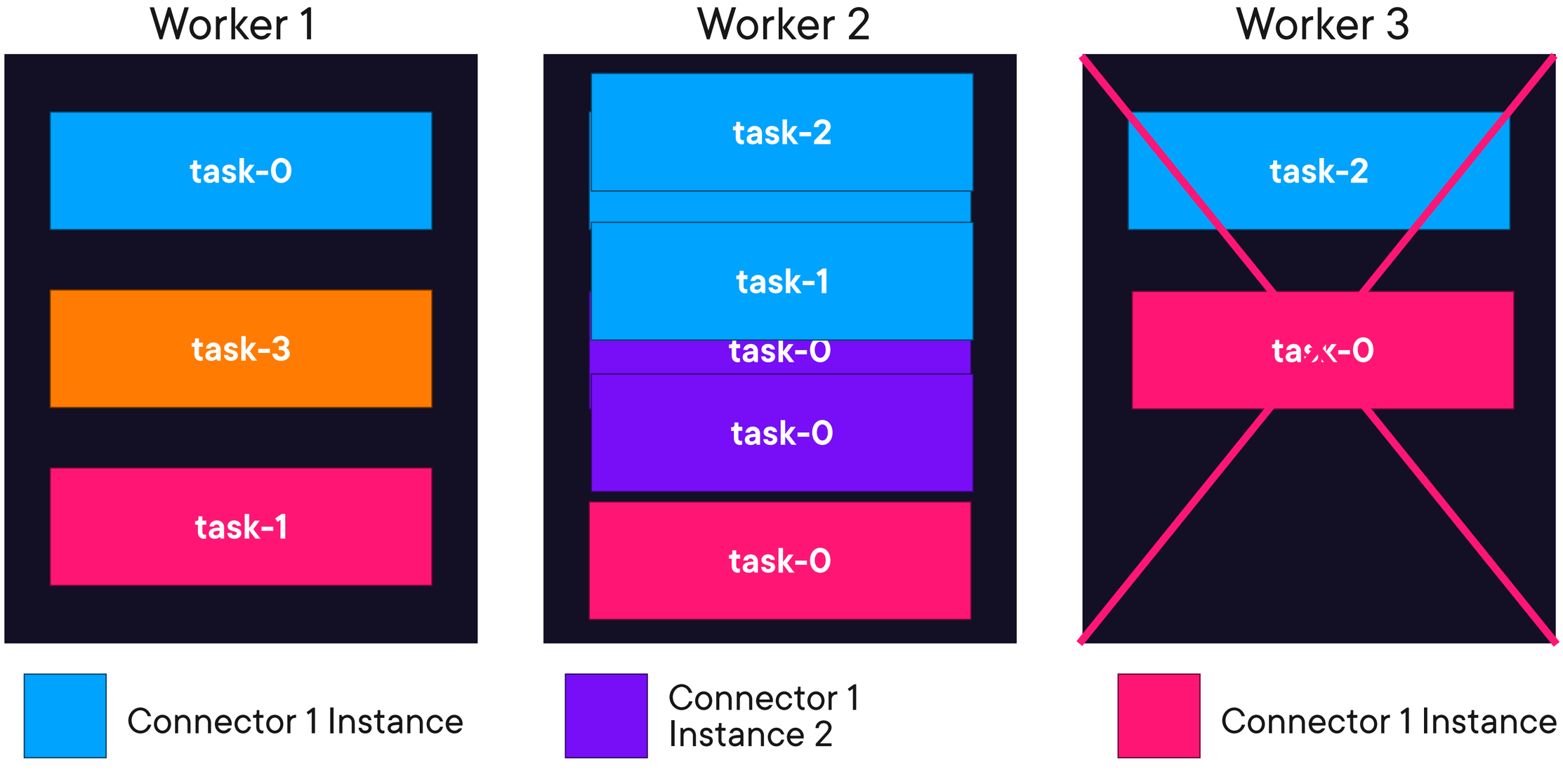
Standalone Example



Standalone Example



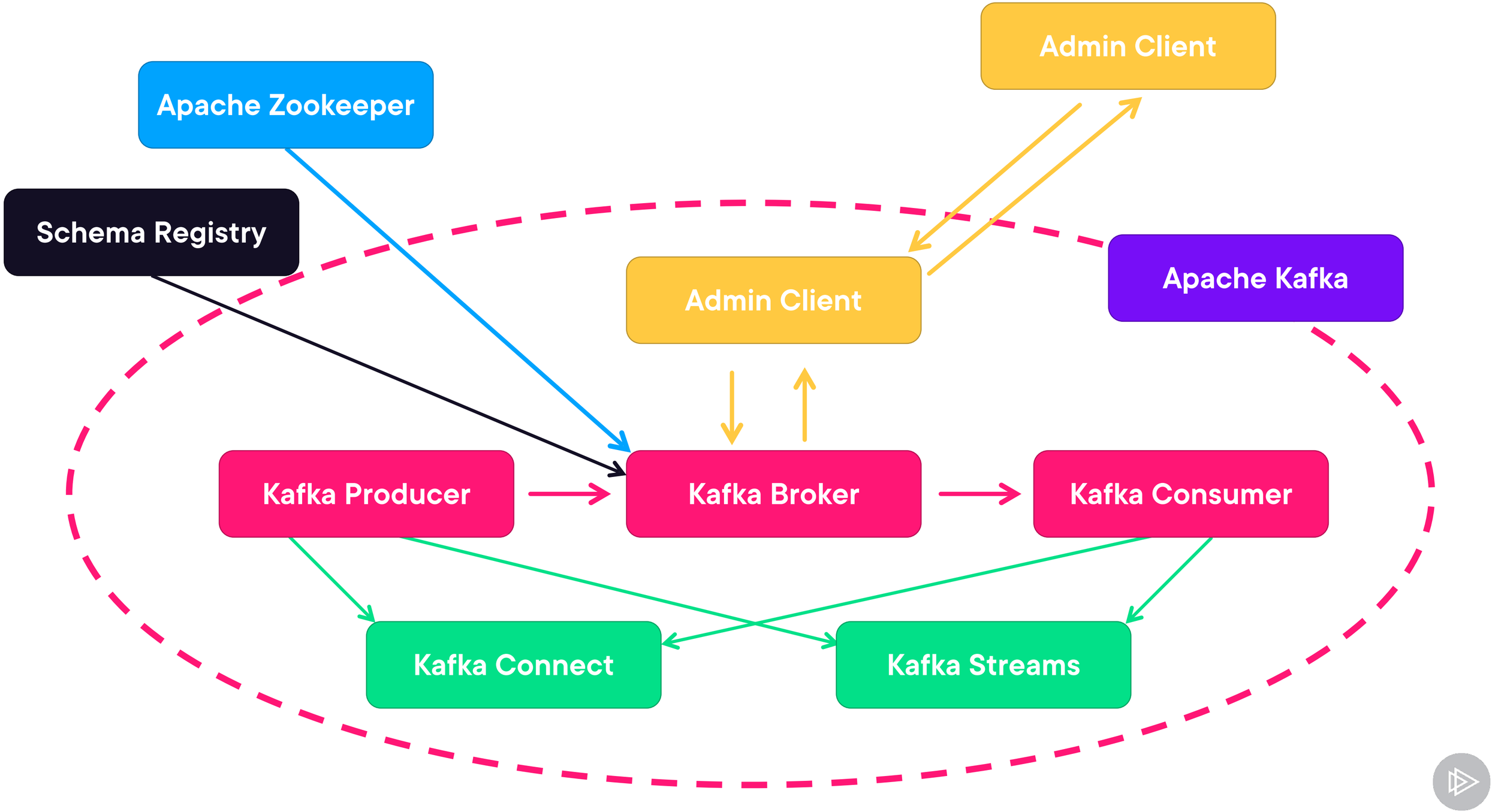
Tasks Rebalancing



Demo

Using the File Connector in a Standalone Mode

Schema Evolution And Enforcement


Admin Client
Apache Zookeeper
Schema Registry
Admin Client
Apache Kafka
Kafka Producer
Kafka Broker
Kafka Consumer
Kafka Connect
Kafka Streams

Avro

Serialization is defined by schema

Schemas are JSON Based

Codegen available at command line

Codegen available by Maven, Gradle, SBT Plugin

Supports the following languages: C, C++, Java, Perl, Python, Ruby, PHP

It Has 3 Modes

Generic mode: Where you use AVRO just to validate it is a valid schema, but don’t do codegen

Specific mode: Where we use an avsc file and the codegen capabilities to generate the code we need

Reflection mode: The inver of the previous mode, based on a class create an avsc file

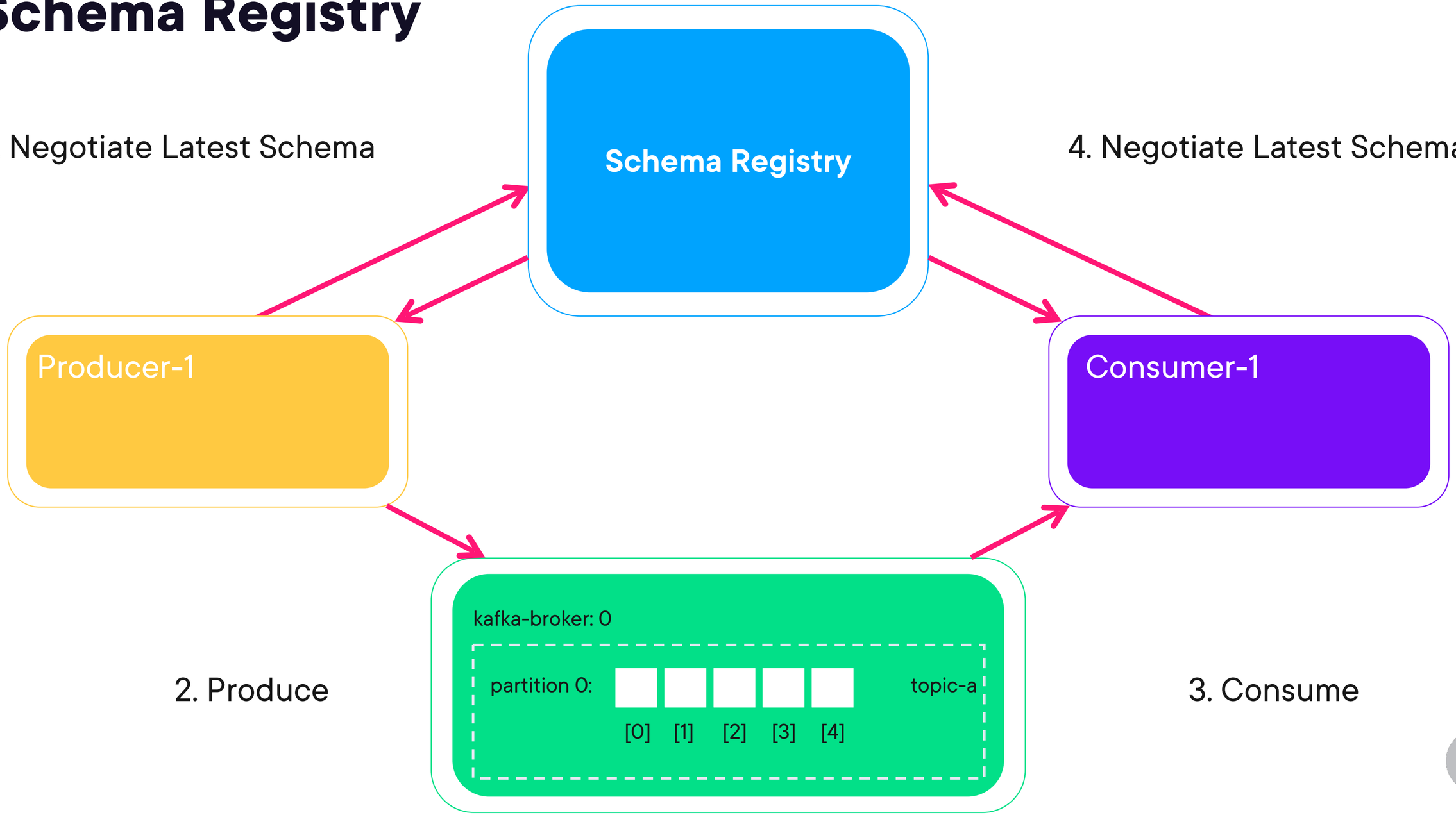
Primitive Types

Avro Type	Java Type
null	null
double	double
float	float
int	int
long	long
bool	bool
string	Unicode CharSequence
bytes	Sequence of 8-bit unsigned bytes

Complex Types

```
{
"namespace“: "com.xyzcorp",
"type“: "record“,
"doc": "An music album“,
"name“: "Album“,
"fields“:[
          {
     "name“: "name“,
     "type“: "string"
          },
     {
     "name“: "yearReleased“,
     "type“: [
     "int“,
     "null"
          ]
        }
     ]
}
```

Schema Registry



```
Properties properties = new Properties();

properties.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,"localhost:9092“);
properties.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,StringSerializer.class);
properties.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,io.confluent.kafka.serializers.
KafkaAvroSerializer.class);
properties.setProperty( "schema.registry.url“, "http://localhost:8081");
```

Schema Registry Java Producer

What makes schema registry work is the Serializer. You must add schema.registry.url and specify the location of the registry.

```
Album album = new Album( "Purple Rain“, "Prince“, 1984,
Arrays.asList( "Purple Rain“, "Let's go crazy“));
ProducerRecord<String, Album> producerRecord =
new ProducerRecord<>( "music_albums“ , "Prince“, album);
producer.send(producerRecord);
```

Example of Sending Custom Avro Object

The ProducerRecord's value is accepting an actual Java Object. We can just send the object just like any other type.

```
Properties properties = new Properties();
properties.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
properties.put(ConsumerConfig.GROUP_ID_CONFIG, "my_group");
properties.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
 "org.apache.kafka.common.serialization.StringDeserializer");
properties.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
 io.confluent.kafka.serializers.KafkaAvroDeserializer.class);
properties.setProperty("schema.registry.url", "http://localhost:8081");
properties.setProperty("specific.avro.reader", "true“ );
```

Schema Registry Consumer

What makes schema registry work is the Deserializer. You must add schema.registry.url and specify the location of the registry. You must add specific.avro.reader and specify that you used specific avro mode.

Receiving a Custom Object from Kafka

```
while (true) {
    ConsumerRecords<String, Album> records =
      consumer.poll(Duration.of(500, ChronoUnit.MILLIS));
    for (ConsumerRecord<String, Album> record : records) {
      System.out.format( "offset: %d\n“ , record.offset());
      System.out.format( "partition: %d\n“ ,
record.partition());
      System.out.format( "timestamp: %d\n“ ,
record.timestamp());
      System.out.format( "timeStampType: %s\n“ ,
      record.timestampType());
      System.out.format("topic: %s\n“, record.topic());
      System.out.format("key: %s\n“, record.key());
      Album a = record.value();
      System.out.format("value: %s\n“, a.getTitle());
      System.out.format("value: %s\n“, a.getArtist());
        }
}
```

Demo

Using the debezium connector with Distributed Kafka Connect to query a topic

Key, Takeaways, and Tips

Takeaways

Kafka Connect can come in two variants: Standalone and Distributed

Kafka connects via worker processes where connectors get deployed as a number of tasks

One can leverage the schema registry and AVRO to send classes through Kafka messages with much more metadata

And thanks to Kafka Connect even send those objects to another system

Keys

Try to deploy a distributed Connect instance that writes to Elasticsearch

Try to Create a full example in Java that uses Avro to send POJOs

Try to ensure everything, in the last demo is clear why we did it that way

Up Next:

Kafka Streams