Kafka Ecosystem

Lesson Objectives

- Learn various tools and components around Kafka
- Confluent stack

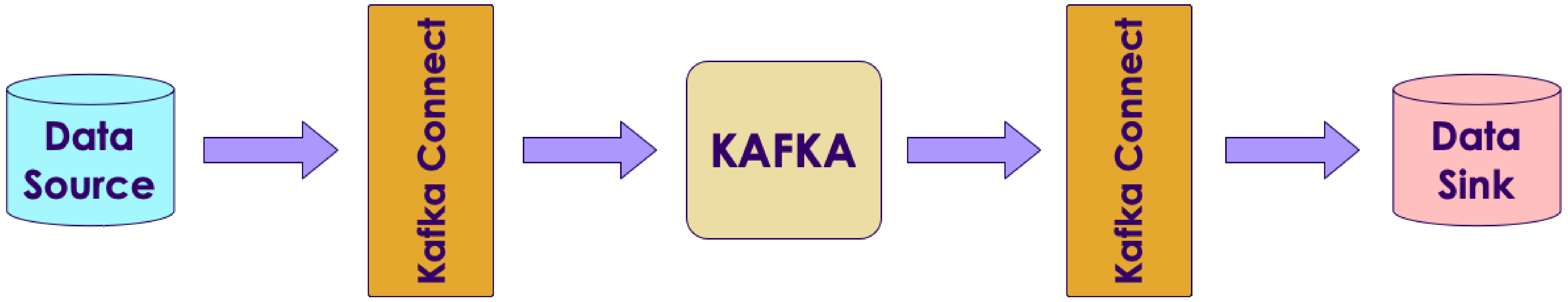
Kafka Eco System

Product	Description
Kafka Streams	Build streaming applications easily
Kafka Connect	Move data between Kafka and other systems (DB / file system)
Kafka Registry	Metadata /schema store for data
Kafka REST Proxy	REST interface into Kafka cluster.,Produce / Consume using RESTFUL APIs
Camus	Kafka / HDFS integration

Kafka Connect

Kafka Connect

- Kafka Connect is a framework included in Apache Kafka that integrates Kafka with other systems.
- It's goal is to make it easy to add new systems to your scalable and secure stream data pipelines.



Kafka Connectors (Supported by Confluent)

Connector	Description	Supported by
ActiveMQ	Source only	Confluent
Amazon S3	Sink	Confluent
Elastic Search	Sink	Confluent
HDFS	Sink (Hadoop, Hive)	Confluent
IBM MQ	Source	Confluent
JDBC	Source and Sink	Confluent
JMS	Source	Confluent

Kafka Connectors (Supported by Vendors)

Connector	Description	Supported by
Azure IoT	Source, IoT	Microsoft
Couchbase	Source and Sink	Couchbase
SAP Hana	Source and Sink	SAP
Vertica	Source and Sink	HP
VoltDB	Sink	VoltDB

Kafka Connectors (Supported by Community)

Connector	Description	Supported by
Amazon Kinesis	Sink (Amazon's managed queue service	Community
Apache Ignite	Source and Sink (File System)	Community
Blockchain	Source Bitcoin,Blockchain	Community
Cassandra	Sink NoSQL	Community
Github	Source	Community
Many more		

Kafka Connect Concepts

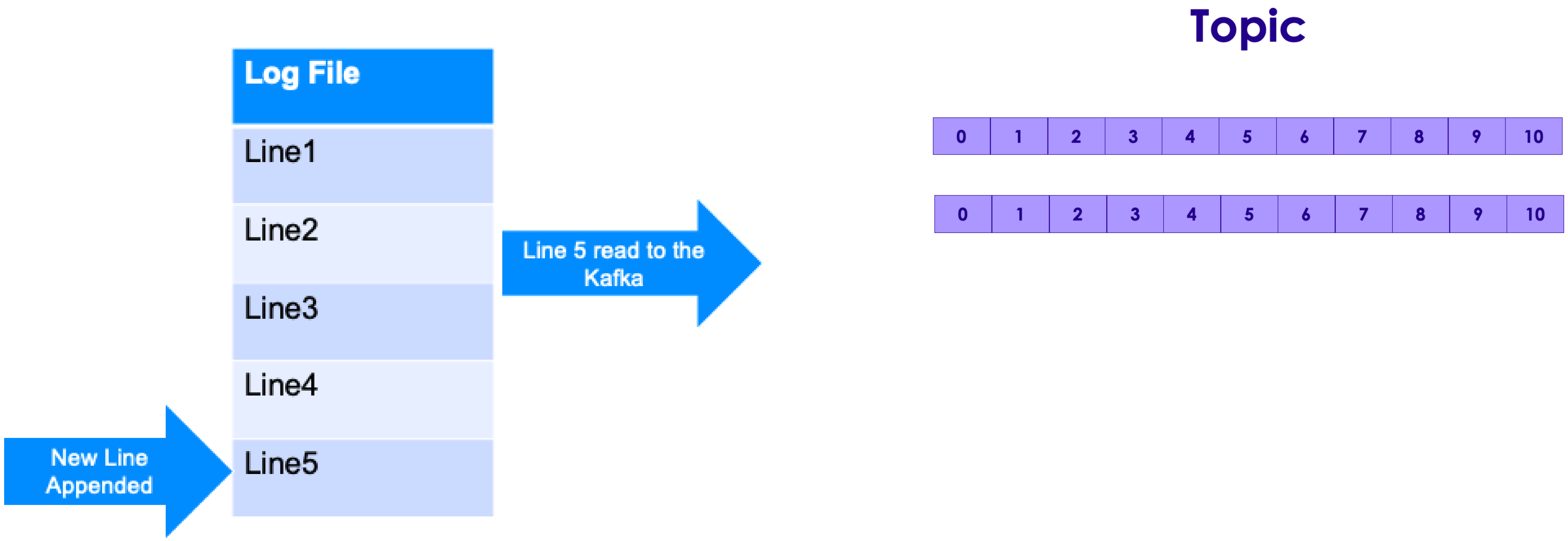
- **Connectors** - A logical process responsible for managing the copying of data between Kafka and another system.
- There are two types of connectors
  - **Source Connectors** import data from another system
  - **Sink Connectors** export data from Kafka
- **Tasks** - Unit of process that handles assigned set of work load by connectors. Connector configuration allows set to maximum number of tasks can be run by a connector.
- **Workers** - Unit of work that schedules connectors and tasks in a process.
- There are two main type of workers: **standalone** and **distributed**

Standalone vs Distributed Workers

- Standalone Worker
  - Single process that executes all connectors and tasks
  - Simple to configure
  - Use for simple use cases or initial testing
- Distributed Worker
  - Provides more scalability and fault tolerance
  - Connectors and tasks are distributed between the workers automatically.

File Connector

- Applications save logs into files on disk
- Streaming logfiles into kafka is a very common use case



File Connector Configuration

- File source connector (File --> Kafka)

```
name=local-file-source
connector.class=FileStreamSource
tasks.max=1
file=/tmp/test.txt
topic=test-topic
```

- File sink connector (Kafka --> File)

```
name=local-file-sink
connector.class=FileStreamSink
tasks.max=1
file=/tmp/test.sink.txt
topics=test-topic
```

- Here is how to run it (in standalone mode)

```
bin/connect-standalone.sh \
    config/connect-standalone.properties \
    config/connect-file-source.properties
```

HDFS Connector

- The HDFS connector exports data from Kafka into HDFS.
- Can also integrate with Hive, so data is readily available for querying using HiveQL
    - Hive tables are partitioned by Kafka topic
- Features
    - Exactly one delivery: Each Kafka message is only exported to HDFS once
    - Supports Avro and Parquet format
    - Secure data transport using Kerberos

HDFS Connector

```
# hdfs-connector.properties

name=hdfs-sink
connector.class=io.confluent.connect.hdfs.HdfsSinkConnector
tasks.max=1
topics=test_hdfs_topic
hdfs.url=hdfs://localhost:9000
flush.size=3
```

- `flush.size` is how many records the connector need to write before invoking file commit
- Running it

```
$  confluent load hdfs-sink -d hdfs-connector.properties
```

- Checking data in HDFS

```
$ hadoop fs -ls /topics/test_hdfs/partition=0
```

```
/topics/test_hdfs/partition=0/test_hdfs+0+0000000000+0000000002.avro
```

Running Connectors in Docker

- It is recommended to run Kafka Connect on containerized environments such as Kubernetes, Mesos, Docker Swarm, or YARN.
- Kafka Connect distributed mode exposes port 8083 by default to serve management REST interface.

```
$ docker run -d \
    --name=kafka-connect \
    --net=host \
    -e CONNECT_BOOTSTRAP_SERVERS="kafka-broker:9092" \
    -e CONNECT_GROUP_ID="group_1" \
    -e CONNECT_CONFIG_STORAGE_TOPIC="kafka-connect-config" \
    -e CONNECT_OFFSET_STORAGE_TOPIC="kafka-connect-offset" \
    -e CONNECT_STATUS_STORAGE_TOPIC="kafka-connect-status" \
    -e CONNECT_KEY_CONVERTER="org.apache.kafka.connect.json.JsonConverter" \
    -e CONNECT_VALUE_CONVERTER="org.apache.kafka.connect.json.JsonConverter" \
    -e CONNECT_INTERNAL_KEY_CONVERTER="org.apache.kafka.connect.json.JsonConverter" \
    -e CONNECT_INTERNAL_VALUE_CONVERTER="org.apache.kafka.connect.json.JsonConverter" \
    -e CONNECT_LOG4J_LOGGERS="io.debezium.connector.mysql=INFO" \
    -v /opt/kafka-connect/jars:/etc/kafka-connect/jars \
    --restart always \
    confluentinc/cp-kafka-connect:3.3.0
```

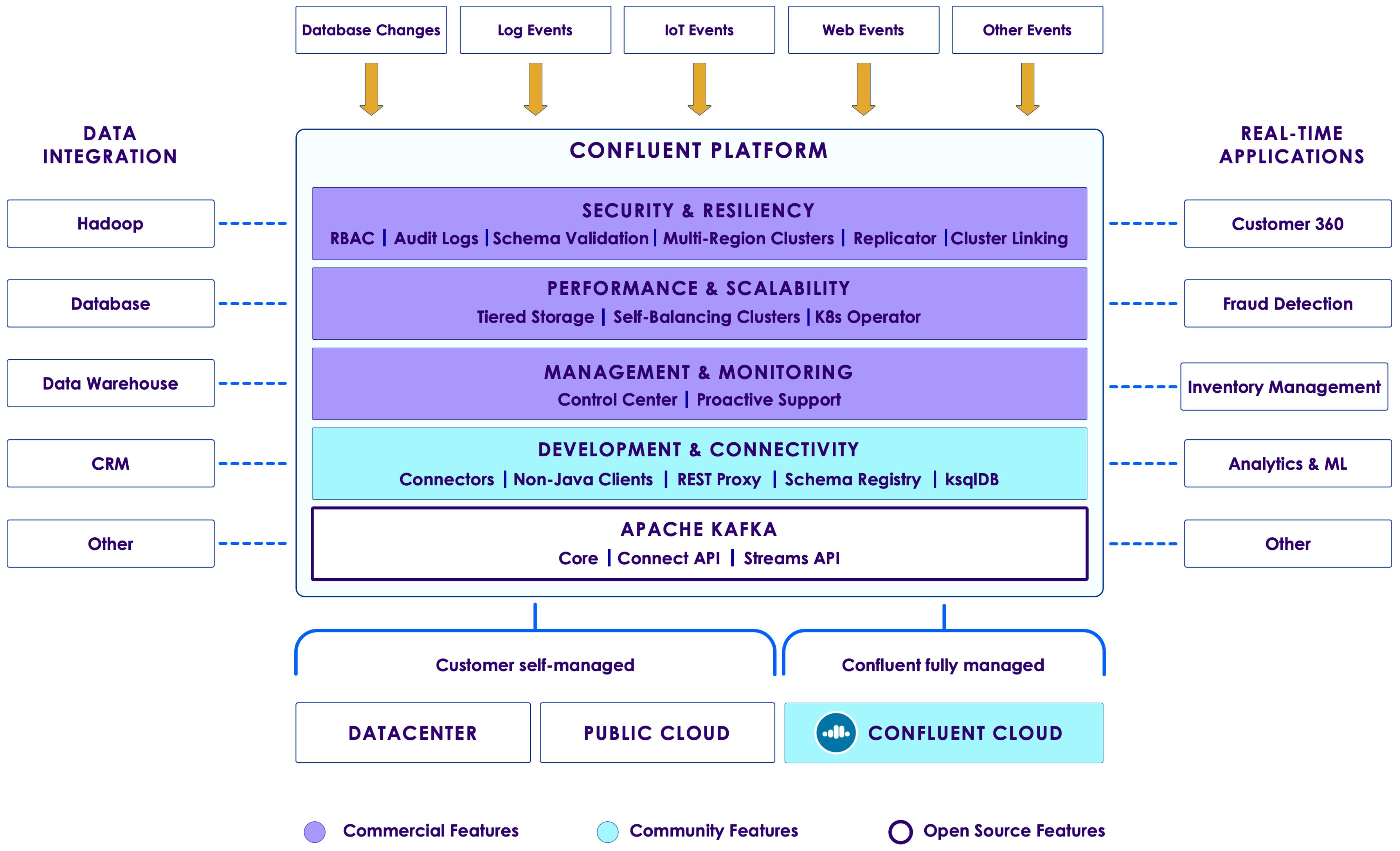
(Optional) Lab: Kafka Connect

- **Overview:**
  - Use Kafka Connect to read data from a file
- **Approximate Time:**
  - 20 - 30 mins
- **Instructions:**
  - Please follow **kafka-connect** lab

Confluent Platform

Confluent Platform

- Confluent platform has the following:
- Free features:
  - Apache Kafka
  - KSQL
  - Connectors (many types of databases)
  - Schema Registry
- Commercial features
  - Control Center
  - Replicator



Lab: Setup Confluent Platform

- **Overview:**
  - Download, install and start Confluent
- **Approximate Time:**
  - 30 - 40 mins
- **Instructions:**
  - CONFLUENT-1-SETUP

Schema Registry

Schemas Evolve in Real Life

- Let's say we have messages in the following format
- Version 1

Id	Type	Success
12345	Click	YES

- Version 2

Id	Type	Success	Message
12345	Click	YES	Page not found

- Q: How will the consumer process this?

Apache Avro

- Data serialization format
- Created for Hadoop project
- Language neutral; can be used from C, Java, Python ..etc
- Schema is described in JSON format
- Data is stored in binary format
- **Supports schema evolutions**
- Use KafkaAvroSerializer
- Apache Avro

Avro Schema

- Version 1

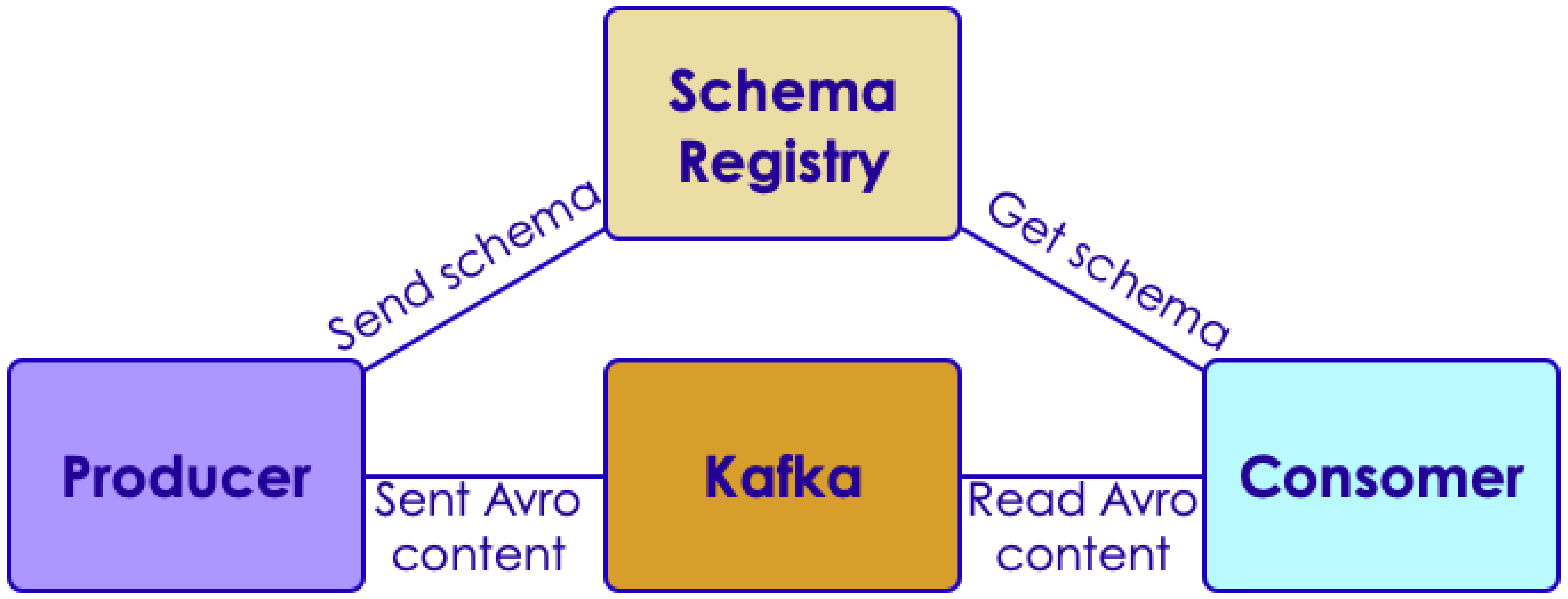
```
{"namespace": "com.example.videos",
  "type": "record",
  "name": "Event",
  "fields": [
     {"name": "id", "type": "int"},
     {"name": "type",  "type": "string"},
     {"name": "success",  "type": "string"}
]
}
```

- Version 2

```
{"namespace": "com.example.videos",
  "type": "record",
  "name": "Event",
  "fields": [
     {"name": "id", "type": "int"},
     {"name": "type",  "type": "string"},
     {"name": "success", "type": "string"},
     {"name": "message", "type": "string"}  // <- new attribute
  ]}
```

Confluent Schema Registry

- Manages schemas and versions
- Provides REST API for interactions
- Works with Kafka seamlessly
- Open-source, downloadable as part of Confluent distribution
- Documentation



Schema Registry Basics

- Schema
    - Structure of an Avro data format
- Subject
    - Scope of the schema. Subject name is derived from topic name by default
    - Schemas can be registered under multiple subjects
        - Automated transparently from Producer as well
- Register a new schema:

```
curl -X POST -H "Content-Type: application/vnd.schemaregistry.v1+json" / \
 --data '{"schema": "{/"type/": /"string/"}"}' / \
 http://localhost:8081/subjects/Kafka-value/versions {"id":1}
```

Schema Registry Examples

- List all schemas under a subject

```
curl -X GET http://localhost:8081/subjects/Kafka-value/versions
```

- Fetch version 1 of the schema

```
curl -X GET http://localhost:8081/subjects/Kafka-value/versions/1

# {"subject":"Kafka-value", "version":1,"id":1,"schema":"\"string\""}`
```

- Register the same schema under a different subject

```
curl -X POST -H "Content-Type: application/vnd.schemaregistry.v1+json" \
  --data '{\"schema\": "{\"type\": \"string\"}"}' http://localhost:8081/subjects/Kafka2-value/versions

# {"id":1}
```

Lab: Use AVRO Schema Registry

- **Overview:**
  - Setup and use Schema Registry
- **Approximate Time:**
  - 30 - 40 mins
- **Instructions:**
  - Avro lab 8.2

Review and Q&A

- Let's go over what we have covered so far
- Any questions?