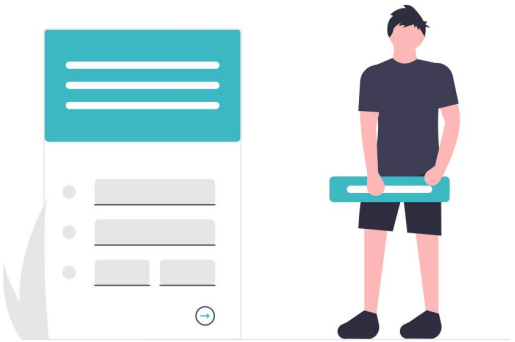

Guillaume C.

mardi 16 novembre 2021

ubik
INGÉNIERIE

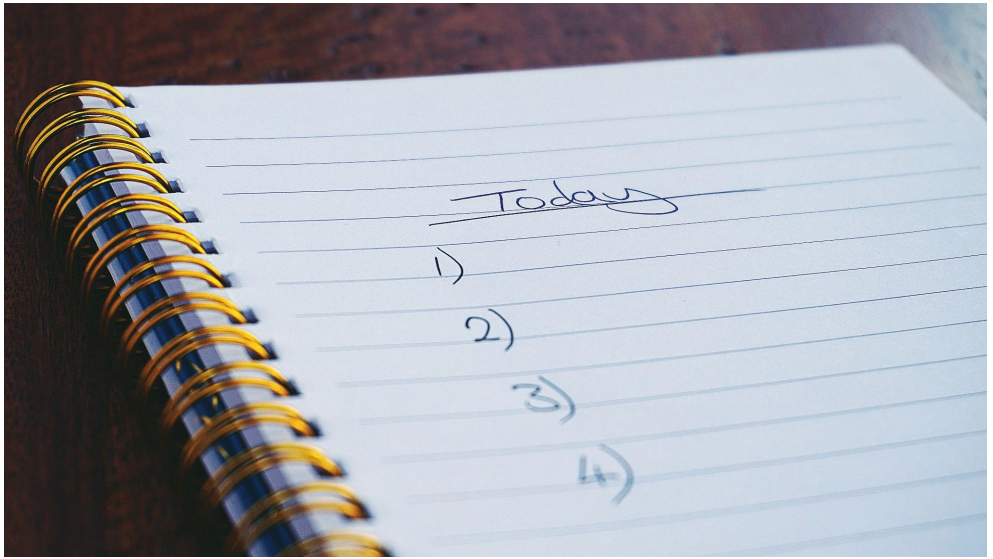
# Qui suis-je ?

- Arrivé à UBIK en Mai 2016
- Formation / Travail sur différents projets (Interne, ADEO, Leroy Merlin)
- Juillet - Décembre 2017 Boulanger (découverte de Kafka)
- Janvier 2018 retour à l'agence ADEO (projet Cookie)
    - Début 2020 intégration de Kafka



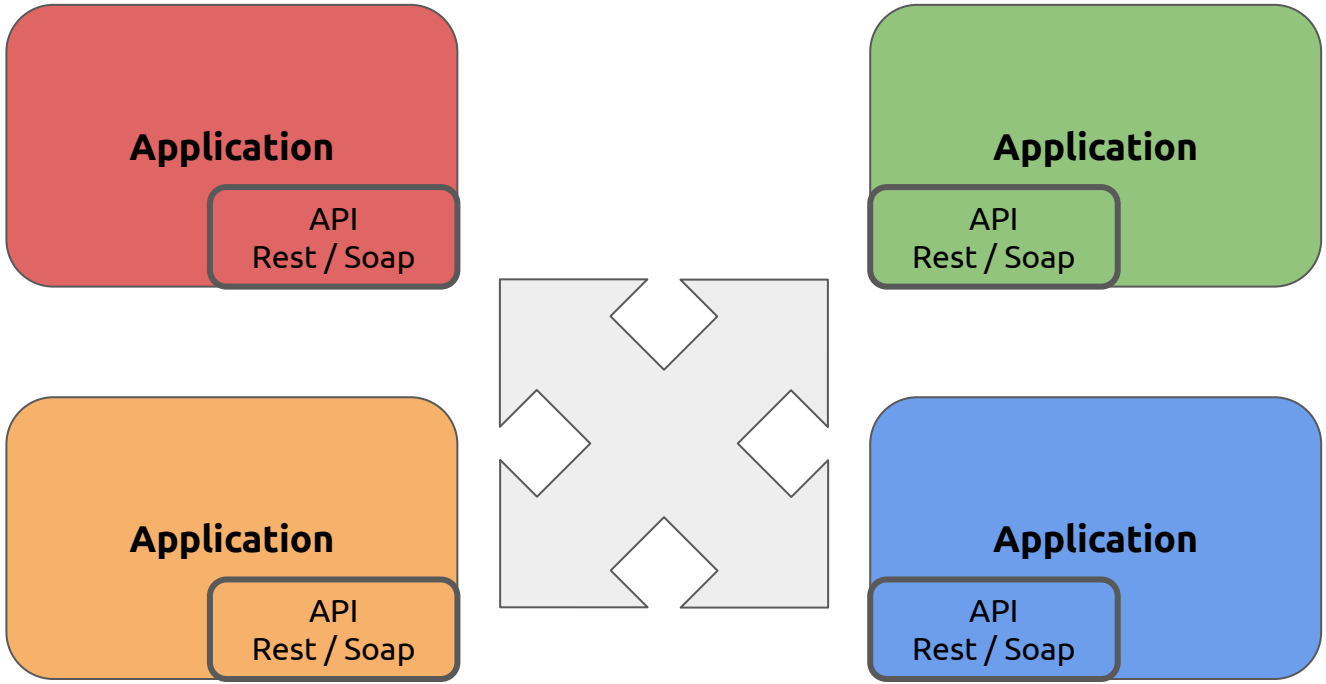
# Plan de la présentation

1. Communication entre applications
2. Kafka, qu’est-ce que c’est ?
3. Concepts
4. Kafka Confluent
5. Intégration/ Démo
6. Retours d’expérience

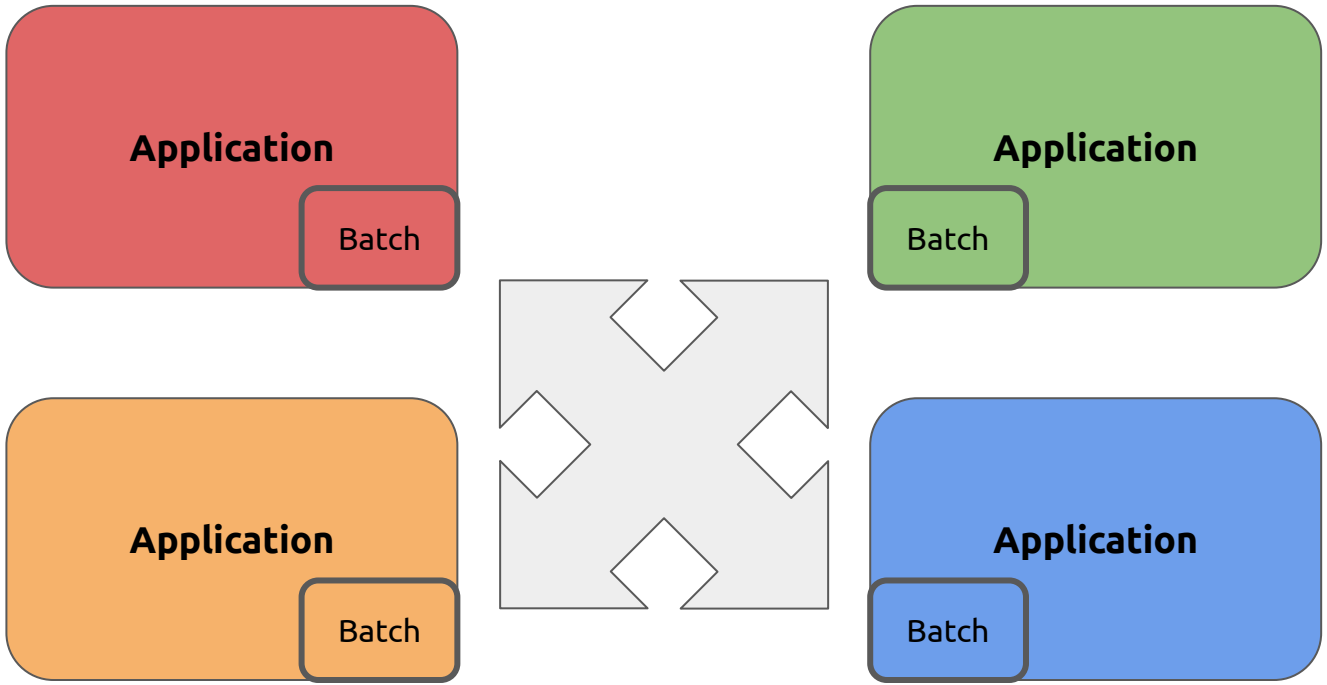


# Communication entre applications

# Communication entre applications



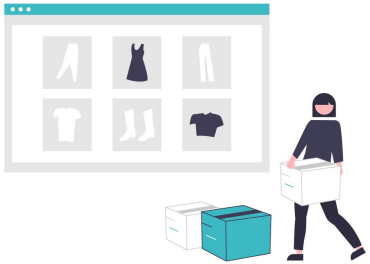
# Communication entre applications



# Communication entre applications

# Kafka, qu’est-ce que c’est ?

# Kafka, qu'est-ce que c'est ?

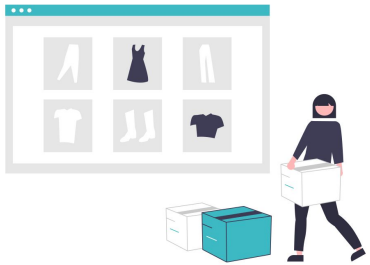


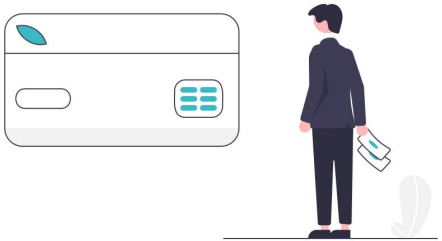
Publication

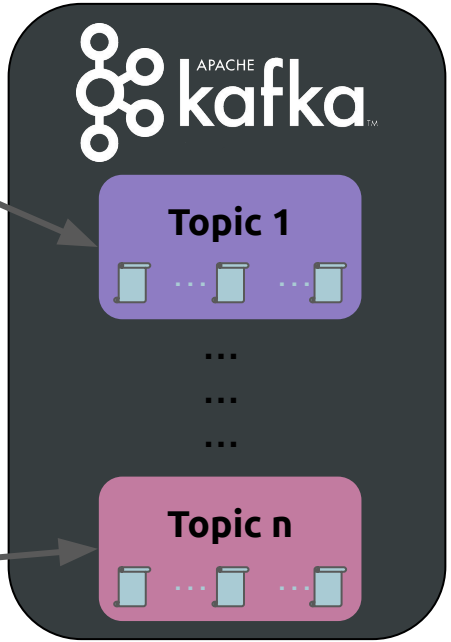

Consommation

# Concepts

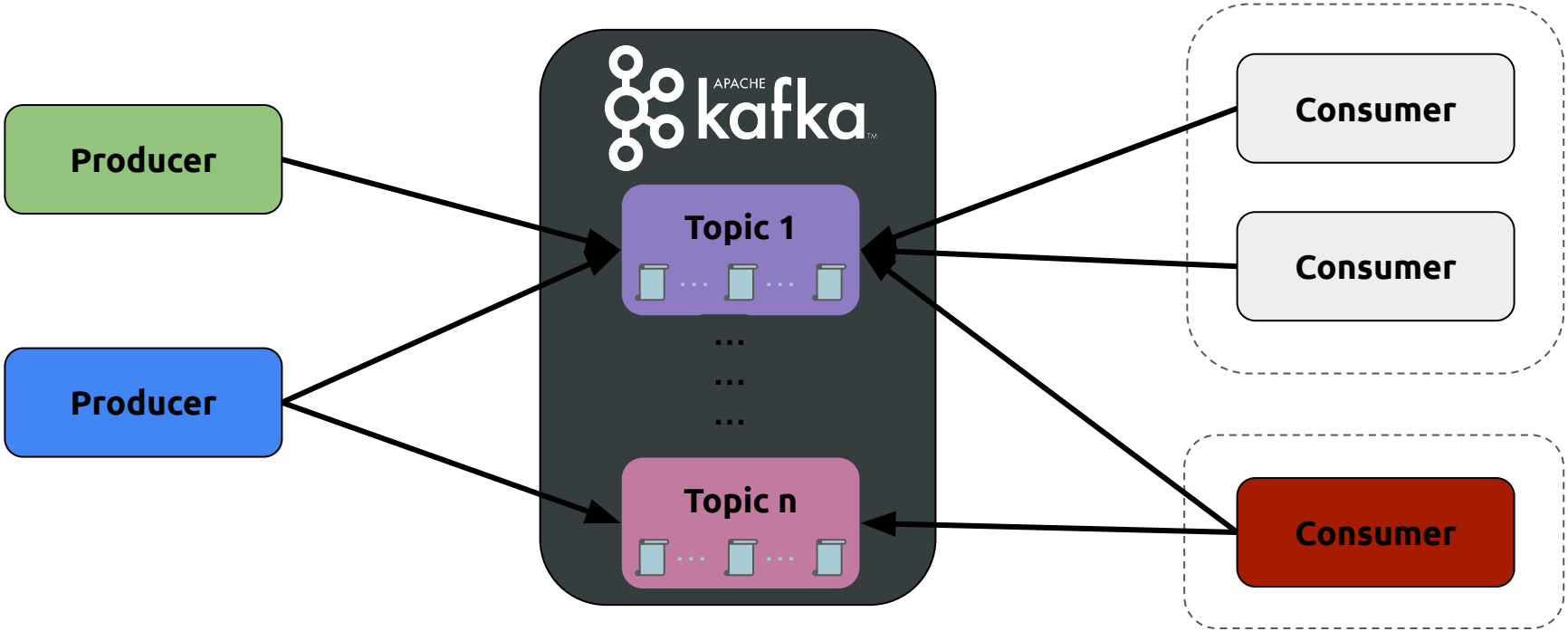
# Topic



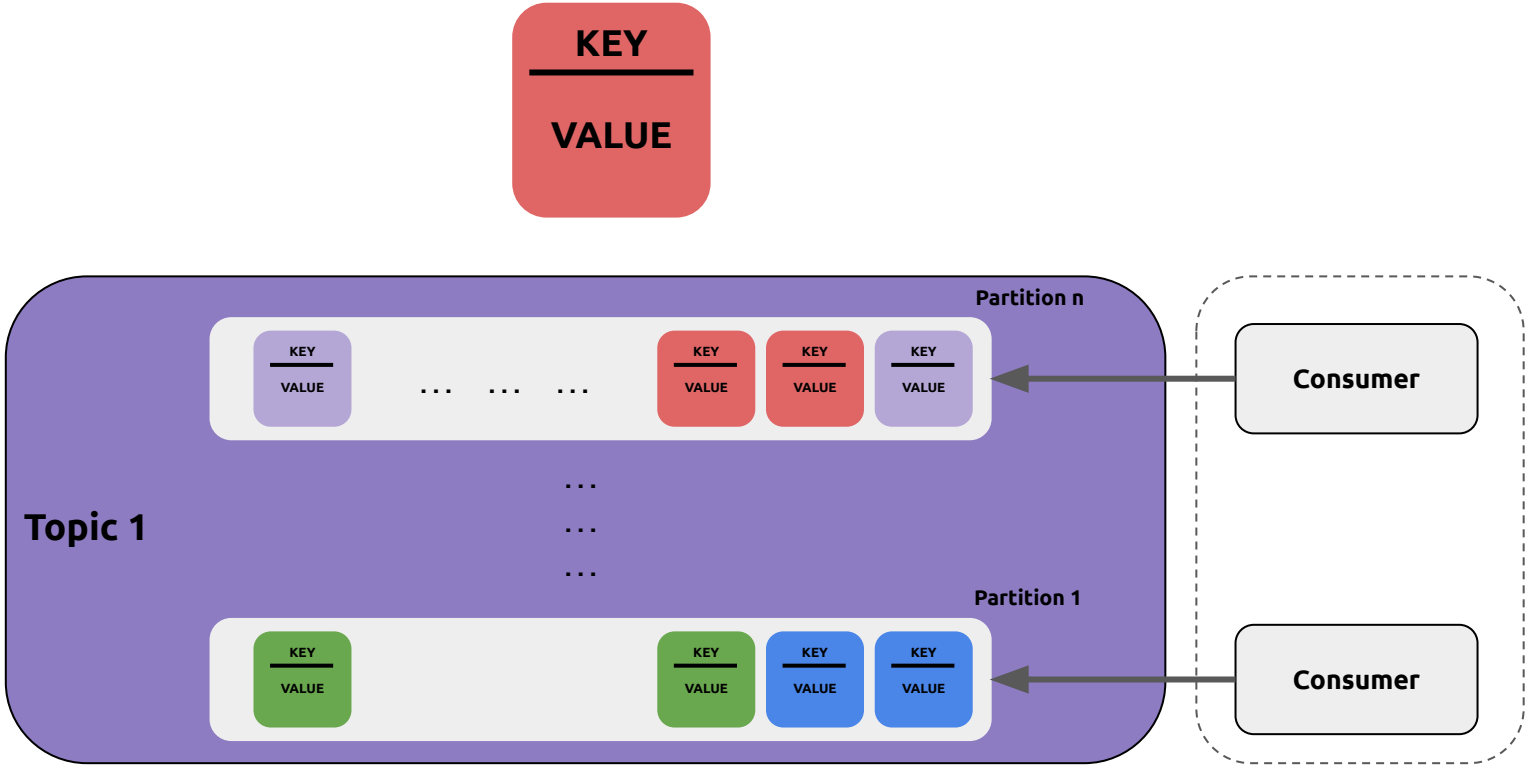




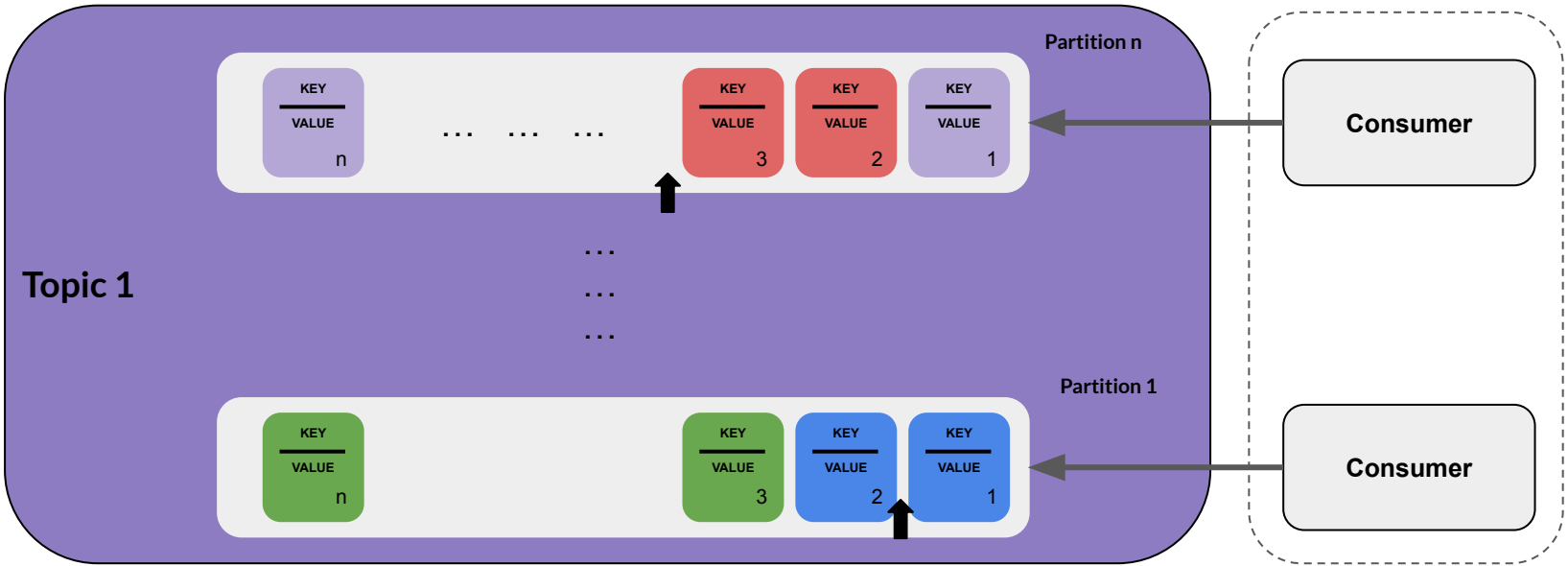

# Producer / Consumer



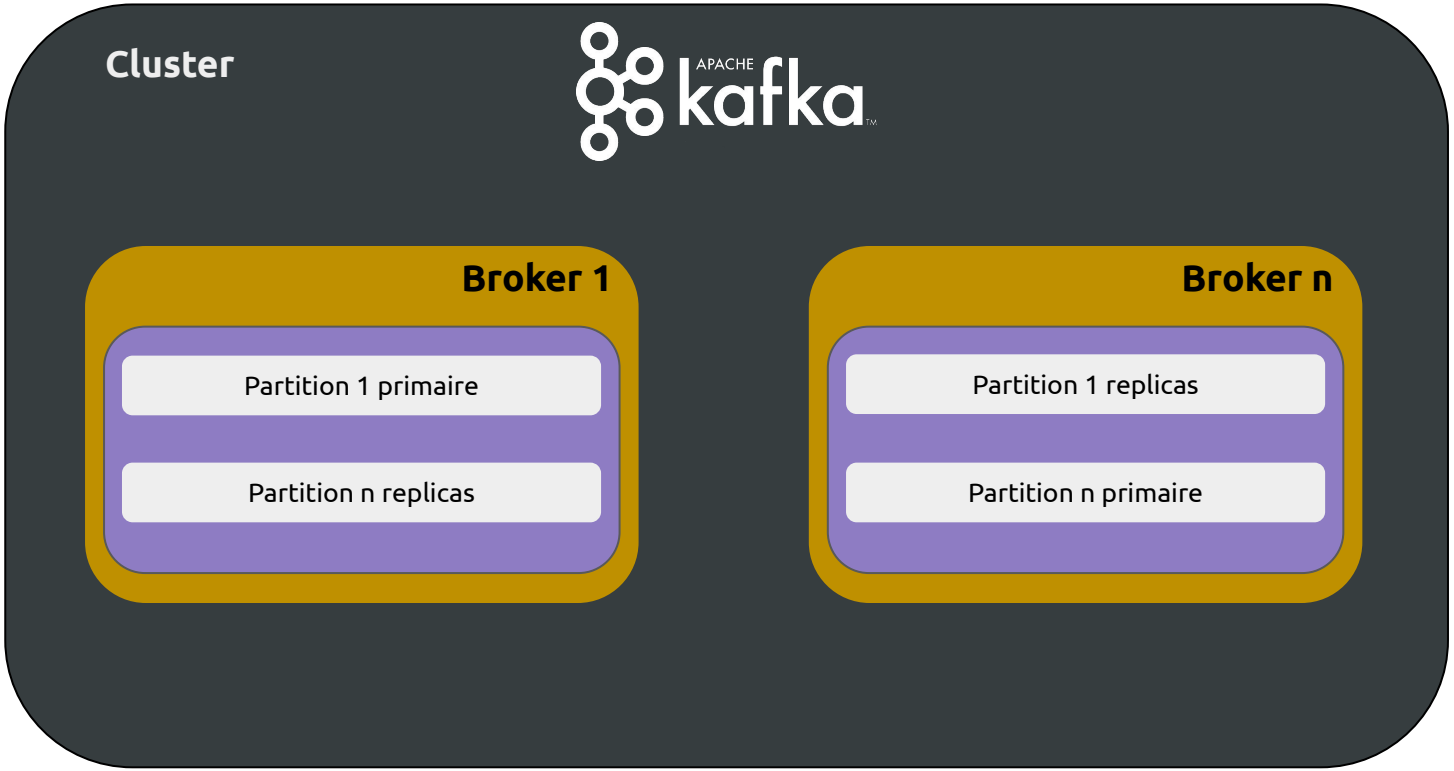
# Partitions


KEY
VALUE
Partition n
KEY
VALUE
KEY
VALUE
KEY
VALUE
KEY
VALUE
Consumer
Topic 1
Partition 1
KEY
VALUE
KEY
VALUE
KEY
VALUE
KEY
VALUE
Consumer

# Offset


Partition n
KEY
VALUE
n
KEY
VALUE
3
KEY
VALUE
2
KEY
VALUE
1
Topic 1
Partition 1
KEY
VALUE
n
KEY
VALUE
3
KEY
VALUE
2
KEY
VALUE
1
Consumer
Consumer

# Brokers / Replicas



# Kafka Confluent

# Kafka Confluent

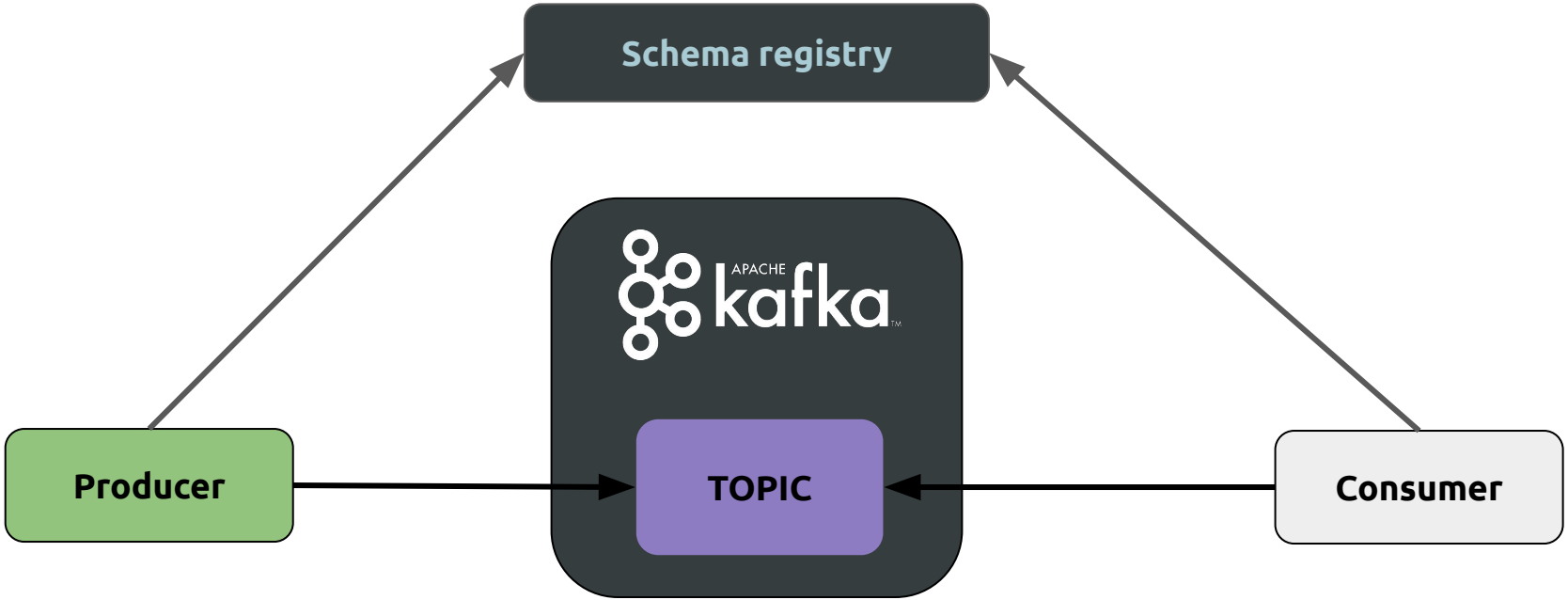
Schema Registry

Kafka Streams

Kafka Connect

ksqlDB

# Schema registry



# Intégration / Démo

# Intégration / Démo

```
<dependency>
  <groupId>org.apache.avro</groupId>
  <artifactId>avro</artifactId>
  <version>${avro.version}</version>
</dependency>

<dependency>
  <groupId>org.springframework.kafka</groupId>
  <artifactId>spring-kafka</artifactId>
</dependency>

<dependency>
  <groupId>io.confluent</groupId>
  <artifactId>kafka-avro-serializer</artifactId>
  <version>${confluent.version}</version>
</dependency>
```

```
spring.kafka.bootstrap-servers
spring.kafka.properties.security.protocol
spring.kafka.properties.sasl.mechanism
…
spring.kafka.consumer.properties.content-type
spring.kafka.producer.key-serializer
```

# Intégration / Démo

## Producer

```
@Autowired
private org.springframework.kafka.core.KafkaTemplate.KafkaTemplate<KeyClass, ValueClass> kafkaTemplate;


public void sendMessage(KeyClass key, ValueClass payload, String outTopic) {

        Map<String, Object> headers = new HashMap<>();
        headers.put(KafkaHeaders.MESSAGE_KEY, key);
        headers.put(KafkaHeaders.TOPIC, outTopic);

        org.springframework.messaging.Message.Message<ValueClass> m = MessageBuilder.withPayload(payload)
                                                                                    .copyHeadersIfAbsent(headers)
                                                                                    .build();
        kafkaTemplate.send(m);
}
```

# Intégration / Démo

## Consumer

```
@KafkaListener(topics = {"${read.topic}"}, autoStartup = "true")
public void processPgwData(@Payload ValueClass message, Acknowledgment acknowledgment) {

}
```

# Retours d’expérience

# Retours d’expérience

- En tant qu’utilisateur, Kafka est une solution simple à intégrer et à utiliser
- Produit stable et performant, quelques millisecondes entre production et consommation, et plusieurs centaines de milliers de messages en quelques secondes
- Ne pas utiliser la dépendance spring-cloud-stream
- En tant que consommateur, bien comprendre qu’il peut y avoir d’autres consommateurs. Gérer un filtre des messages
- Attention au blocage de partition si erreurs lors de la consommation
    - un message non consommé bloque totalement la partition
    - déplacer l’offset nécessite de stopper tous les consommateurs du groupe
- Ré-authentification n’est pas automatique dans la config SpringBoot
- Deserialiser Avro org.apache.avro.util.Utf8.Utf8 et non String

# Des questions ?