



Apache Kafka

Using OpenEdge Messaging

Fadi Nassereddine
Software Engineer 2

November 15 2023





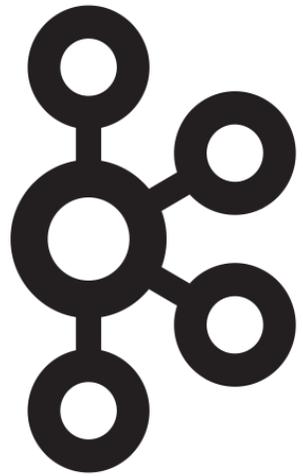
I cannot make you understand. I cannot make anyone understand ... I cannot even explain it to myself.

Franz Kafka

The Metamorphosis



What is Apache Kafka?



kafka



Apache Kafka is an open-source distributed **event streaming** platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

<https://kafka.apache.org/>

Kafka Event Streaming



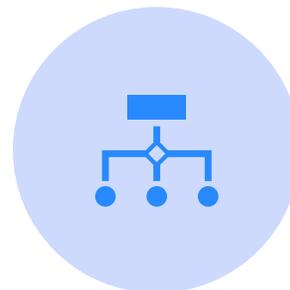
Publish (write) streams of events



Subscribe to (read) streams of events



Store streams of events



Process streams of events
(as they occur or later)

Some Use Cases



Messaging

Kafka can work as a replacement for message brokers such as SonicMQ



Log Aggregation

Centralized storage and processing of logs as a stream of messages.



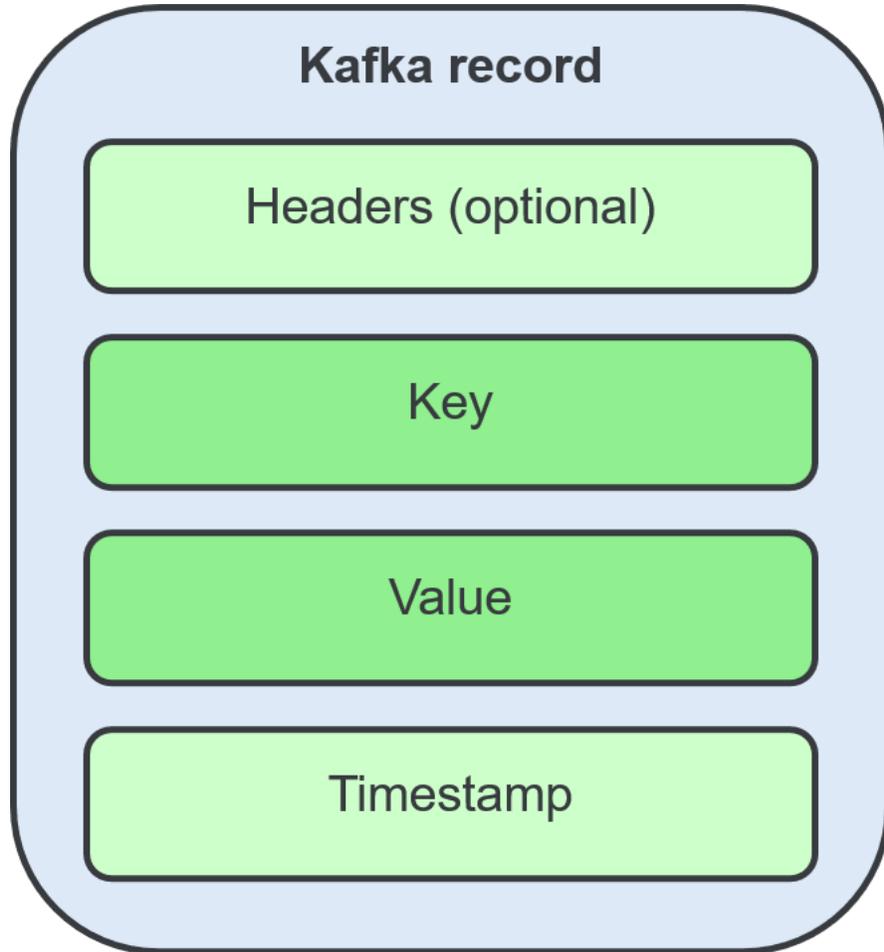
Data Processing

Low-latency transmission of data to consumers. Useful for financial data, predictive maintenance, logistics, etc.



Activity Tracking

The original use case for Kafka. Clicks, registrations, likes, orders, preferences, etc.



In Apache® Kafka®, events are also called messages or records.

Concepts

Event – key: Mike, value: 5€ payment to Cameron, timestamp: Sep 1, 2022

Producer – client application that writes events to Kafka

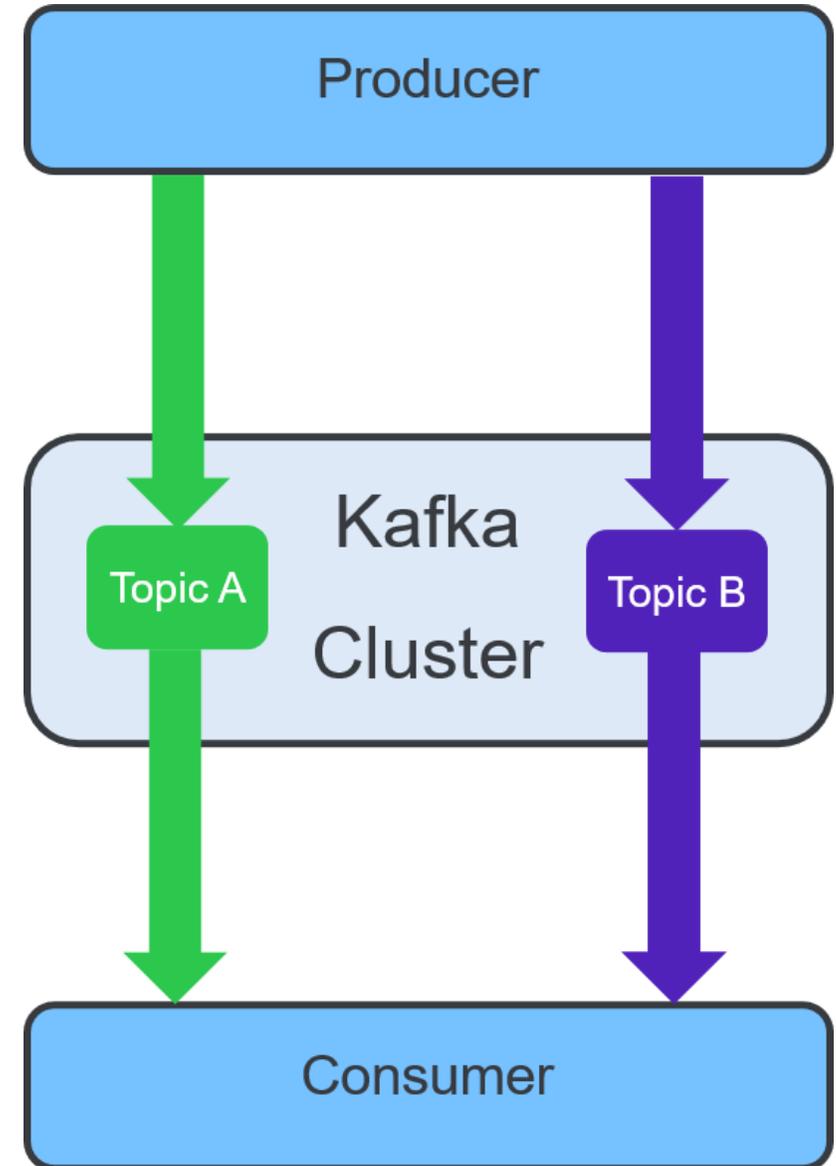
Consumer – applications that read and process events from Kafka

Topic – a virtual “folder” of events

Partition – Topic storage is spread over some number of brokers

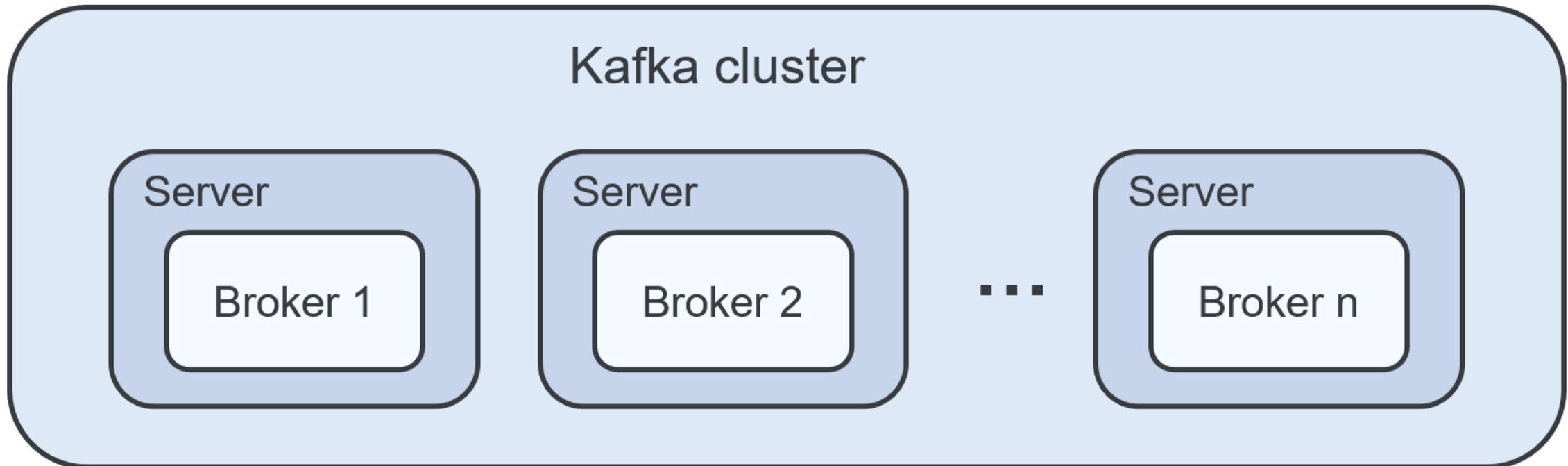
Topics Are Related Messages

- Similar to a log – a sequence of messages
- One or many producers can write to one or many topics
- Message streams are persistent. Kafka can be configured to retain messages for a certain length of time or based on storage size.

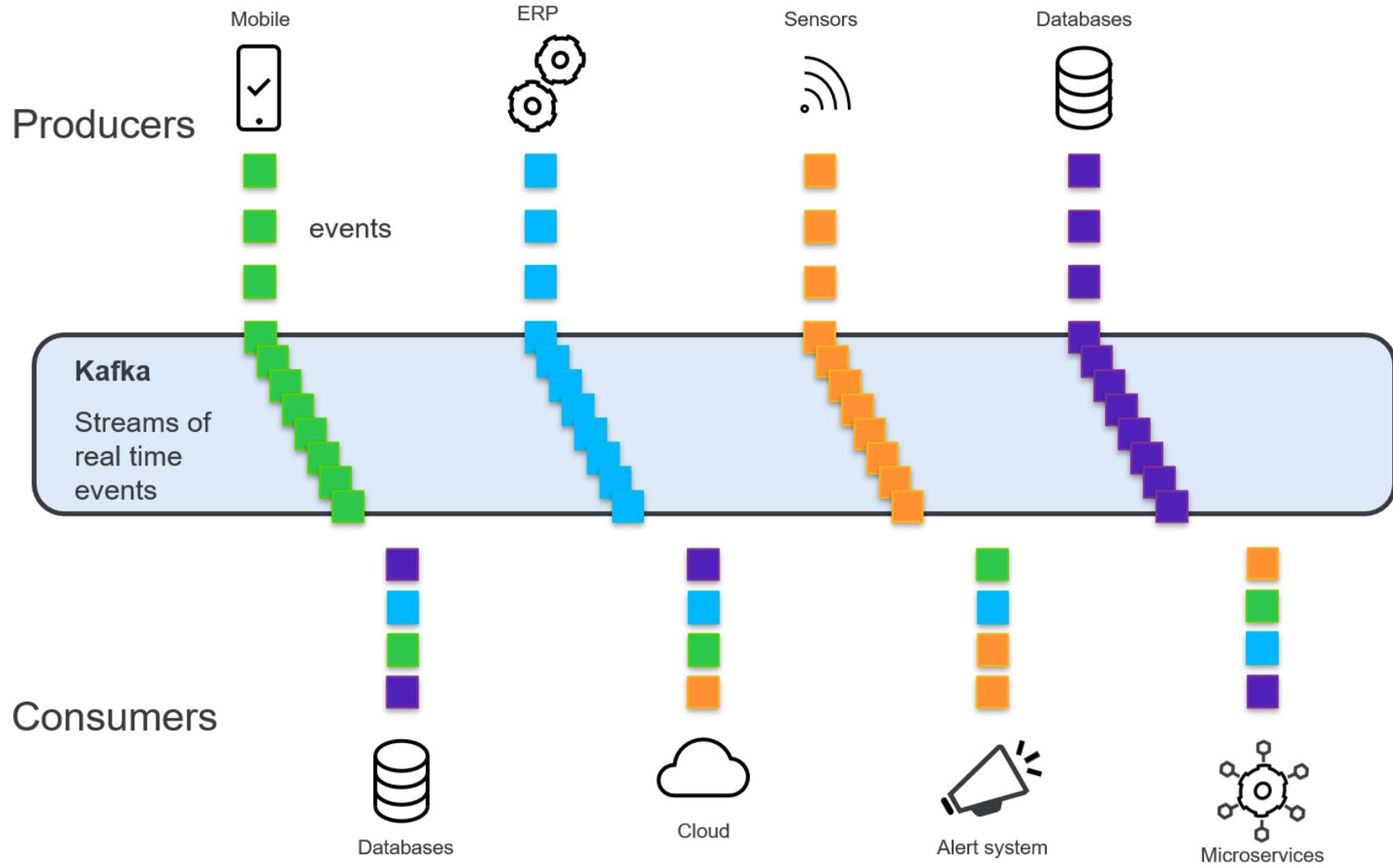


Kafka Clusters and Brokers

- Kafka is designed to be immensely scalable
- Brokers are split across multiple servers and grouped into clusters

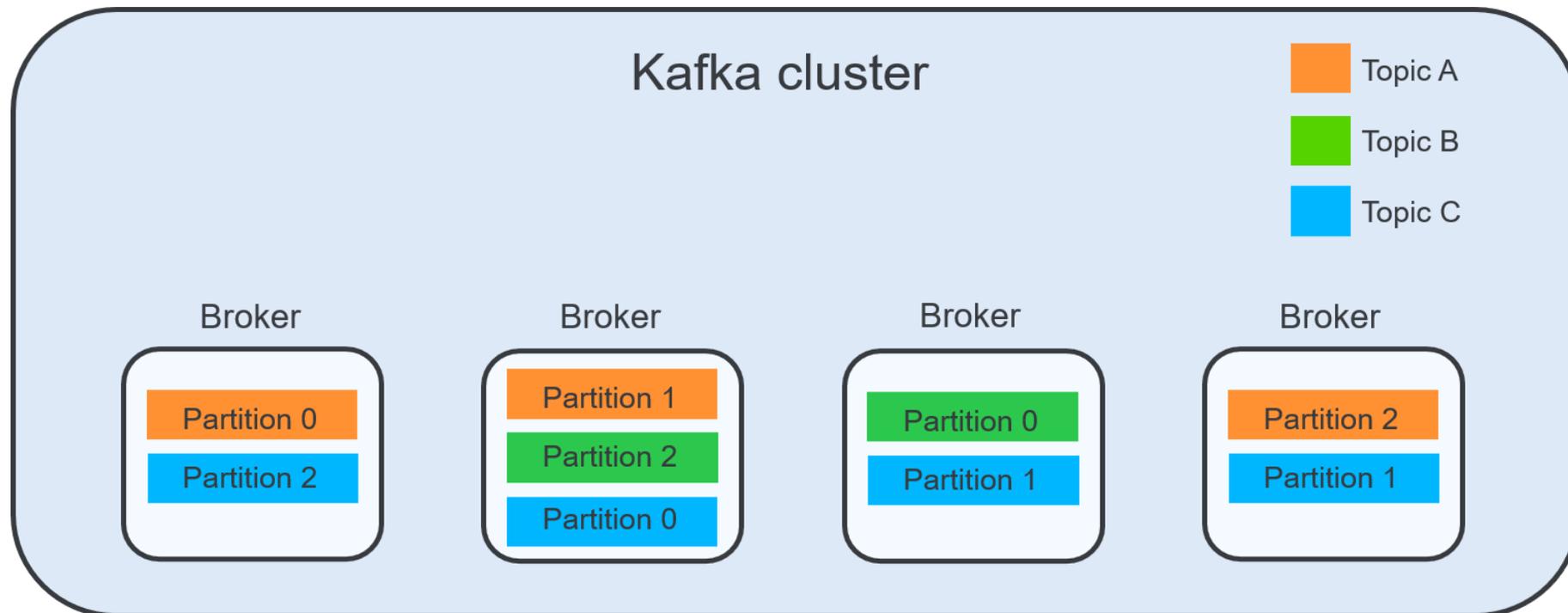


Producers and Consumers



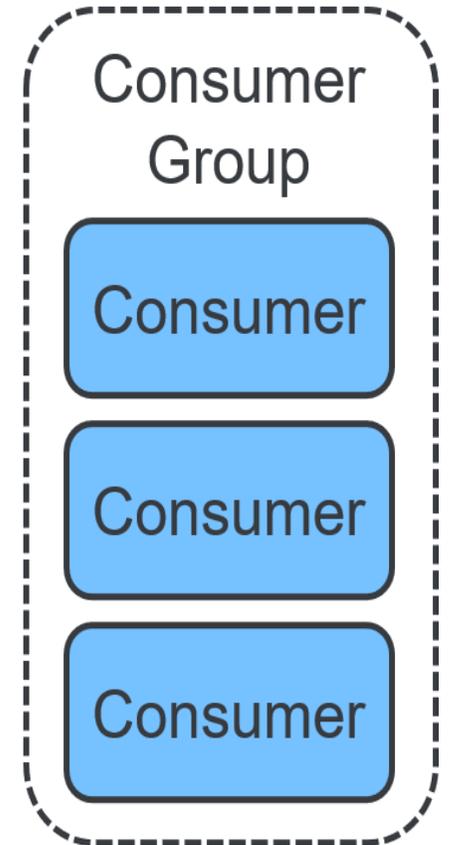
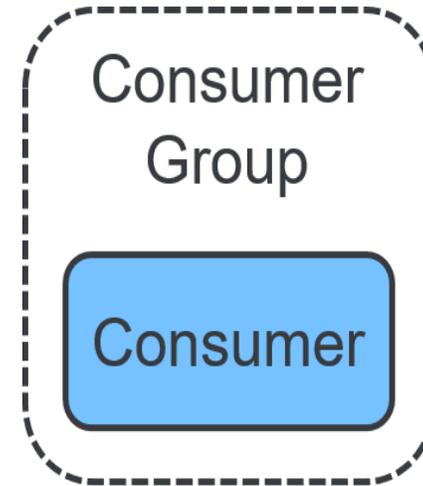
Scaling via Partitions

- Message streams can be distributed across brokers
- Can be round-robin, by partition number, or hash of key
- For safety, topics can be replicated between brokers

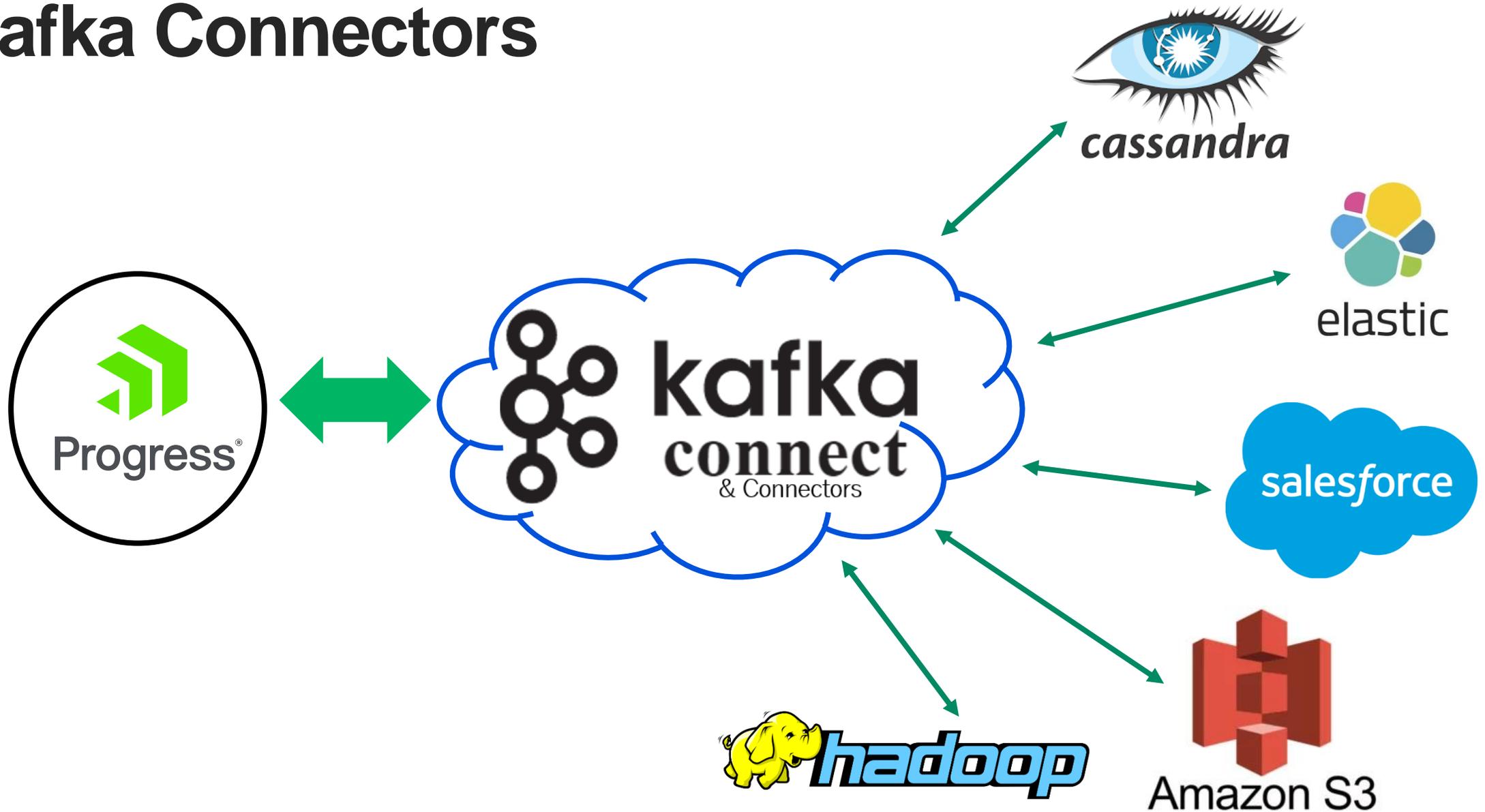


Consumer Groups

- A group of clients acting together to consume messages
- Distributes the load between clients
- OpenEdge consumers must be part of a group



Kafka Connectors





How can the ABL use Kafka?

OpenEdge Messaging 12.5+

- New feature/API in ABL
- Modern Object-Oriented API for supporting messaging providers
- Can be called from procedural ABL
- Initial support is for Kafka, but API not tied to Kafka (future transports)
- Uses common OO patterns such as Builders
- Defines Kafka Producer and Consumer

Parent package: OpenEdge

OpenEdge.Messaging

Subpackages

 OpenEdge.Messaging.Kafka

Interfaces

-  OpenEdge.Messaging.IConsumer
-  OpenEdge.Messaging.IConsumerRecord
-  OpenEdge.Messaging.IDeserializationContext
-  OpenEdge.Messaging.IDeserializer
-  OpenEdge.Messaging.IHeaders
-  OpenEdge.Messaging.IProducer
-  OpenEdge.Messaging.IProducerRecord
-  OpenEdge.Messaging.ISendResponse
-  OpenEdge.Messaging.ISerializationContext
-  OpenEdge.Messaging.ISerializer
-  OpenEdge.Messaging.ITopicConfiguration

Classes

-  ConsumerBuilder
-  ConsumerConfig
-  ConsumerRegistry
-  Headers
-  JsonSerializer
-  JsonSerializer
-  MemptrDeserializer
-  MemptrSerializer
-  MessagingError
-  ProducerBuilder
-  ProducerConfig
-  ProducerRecord
-  ProducerRegistry
-  RecordBuilder
-  RecordBuilderConfig
-  RecordHeader
-  SendResponse
-  StringDeserializer
-  StringSerializer
-  TopicConfiguration
-  TopicConfigurationBuilder

OpenEdge.Messaging.Kafka

Interfaces

 OpenEdge.Messaging.Kafka.IConfigBuilder

Classes

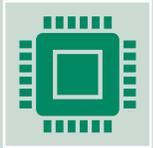
-  Acks
-  AutoOffsetReset
-  ConsumerBuilderDelegate
-  IsolationLevel
-  KafkaClientBuilderUtil
-  KafkaClientConfig
-  **KafkaConsumer**
-  KafkaConsumerBuilder
-  KafkaConsumerConfig
-  KafkaConsumerDelegate
-  KafkaConsumerRecord
-  KafkaDeserializationContext
-  **KafkaProducer**
-  KafkaProducerBuilder
-  KafkaProducerConfig
-  KafkaRecordBuilder

Using Kafka with OpenEdge



Install the Apache Kafka library, **librdkafka**.

<https://docs.progress.com/bundle/openedge-kafka-guide/page/How-to-Set-Up-Apache-Kafka-with-Progress-OpenEdge.html>



Add the relevant OpenEdge libraries to your PROPATH.

`$DLC/tty/netlib/OpenEdge.Net.pl` (HTTP)
`$DLC/tty/messaging/OpenEdge.Messaging.pl`
(OpenEdge Messaging)



Use the ABL-Kafka Producer and Consumer APIs.

OpenEdge Integration with Kafka

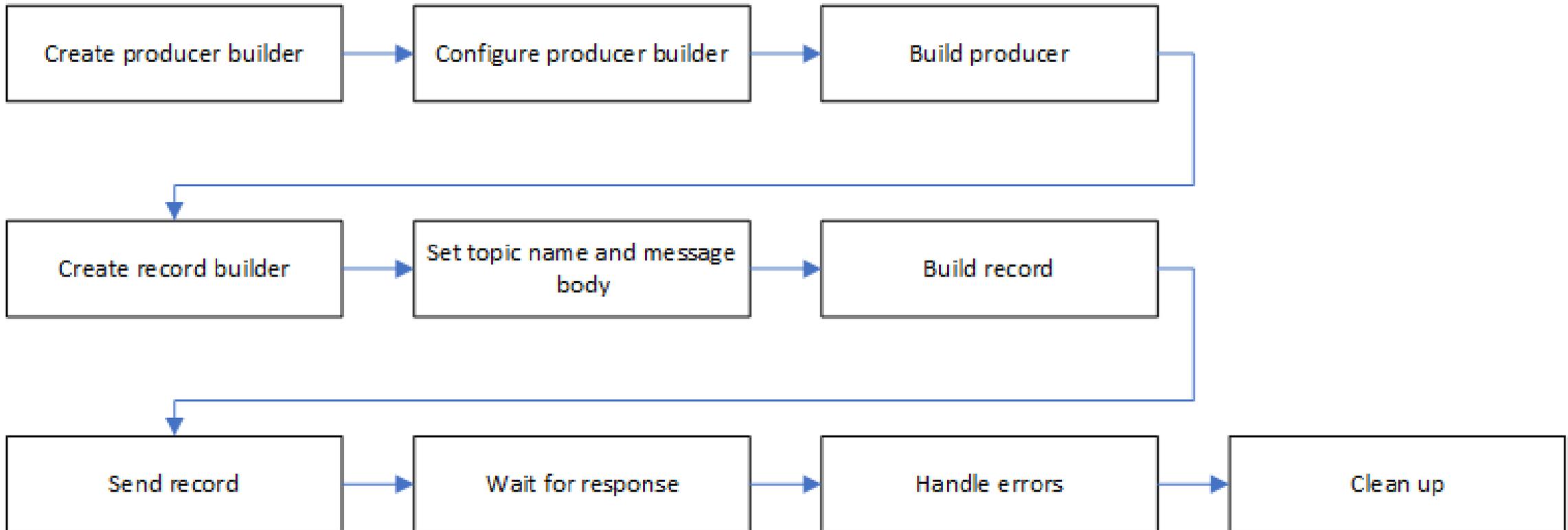
- Everything configurable through name=value string pairs.
- Support for transactional messaging 12.6+
- Support for client-side Quality-of-Service configuration
 - All the networking, response handling, error handling is handled by `librdkafka` and OpenEdge.
- Logging is integrated with OpenEdge client log file.
- Application errors are handled via ABL thrown errors and status objects.



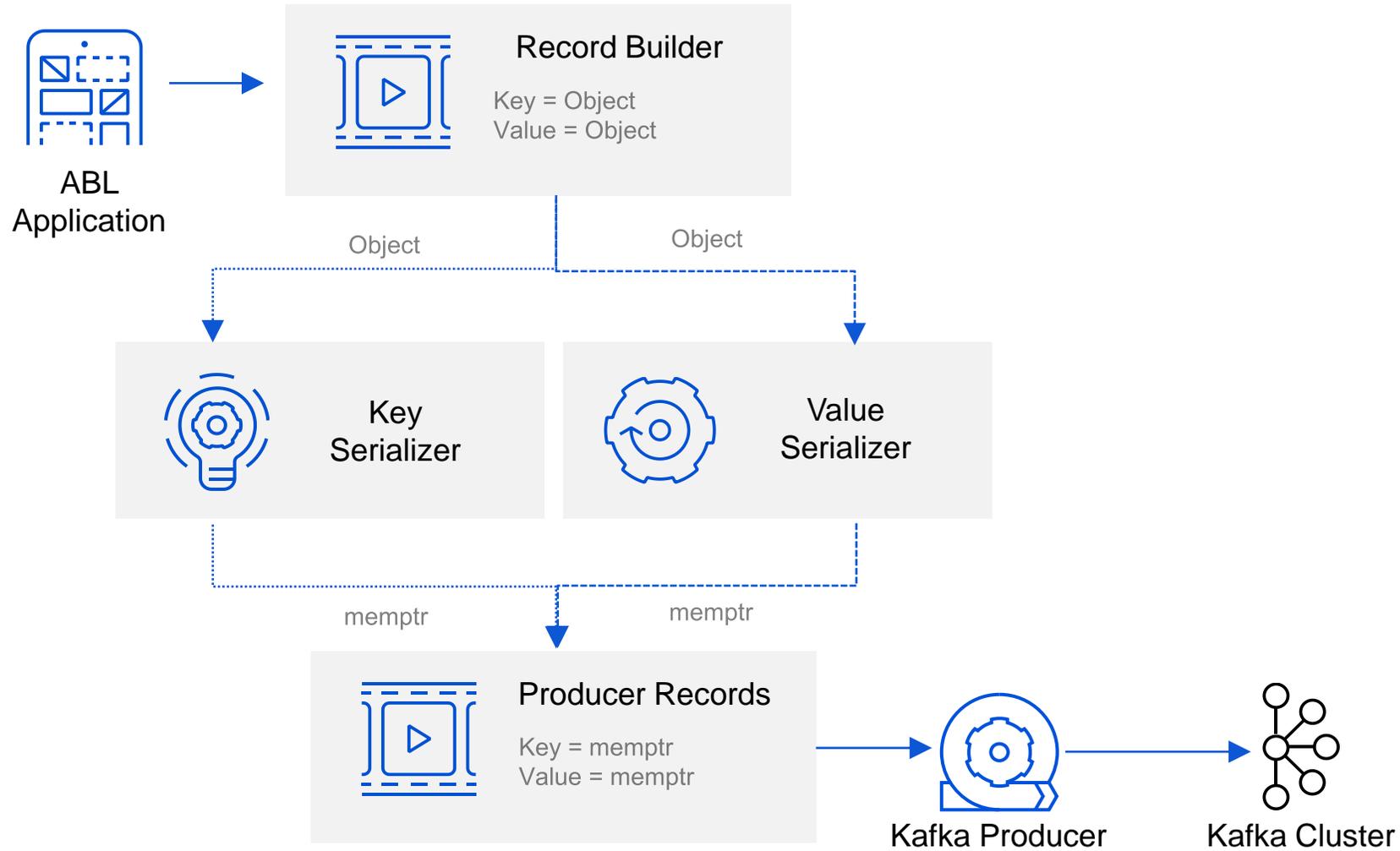
ABL Can Send Kafka Messages

OpenEdge Producer

- Creates messages (records/events)
- Sends them to a Kafka cluster



Producer Message Flow



Serializers

- Custom serializers
- Provided serializers

Serializer	Description
OpenEdge.Messaging.StringSerializer	Simple string-base text. Converts strings to bytes and uses UTF-8 character encoding.
OpenEdge.Messaging.MemptrSerializer	Pass-through serializer.
OpenEdge.Messaging.JsonSerializer	Converts ABL JSON objects to bytes.

```
pb:SetBodySerializer( new StringSerializer() ).  
pb:SetKeySerializer( new StringSerializer() ).
```

Message Producer

```
var OpenEdge.Messaging.RecordBuilder recordBuilder.  
var OpenEdge.Messaging.Kafka.KafkaProducerBuilder pb  
var OpenEdge.Messaging.IProducer producer.  
var OpenEdge.Messaging.IProducerRecord record  
var OpenEdge.Messaging.ISendResponse response
```

```
pb = cast(OpenEdge.Messaging.Kafka.KafkaProducerBuilder, OpenEdge.Messaging.Kafka.KafkaProducerBuilder).  
pb: SetBootstrapServers("localhost:9092")  
pb: SetBodySerializer(new JsonSerializer()).  
producer = pb: Build().
```

```
recordBuilder = recordBuilder: SetKey("key").  
recordBuilder: SetValue("message body").  
record = recordBuilder: Build().  
response = producer: Send(record).
```

Create the ProducerBuilder using "kafka"

Create a serializer for the value

Specify a Kafka cluster server

Build the producer

Give the topic to which you want to publish

Add a message value

Send the message to the Kafka cluster!

Sending messages from the ABL

- Producer usable from any GUI, CHUI, Batch, and Progress Application Server
- Messages are always sent async
- Response objects used by application to check status of sent message
- Response updates are done in the background by the ABL client
 - Coordinated with garbage collection, PAUSE or Flush()
- ABL application determines *if- and-when* to check the responses

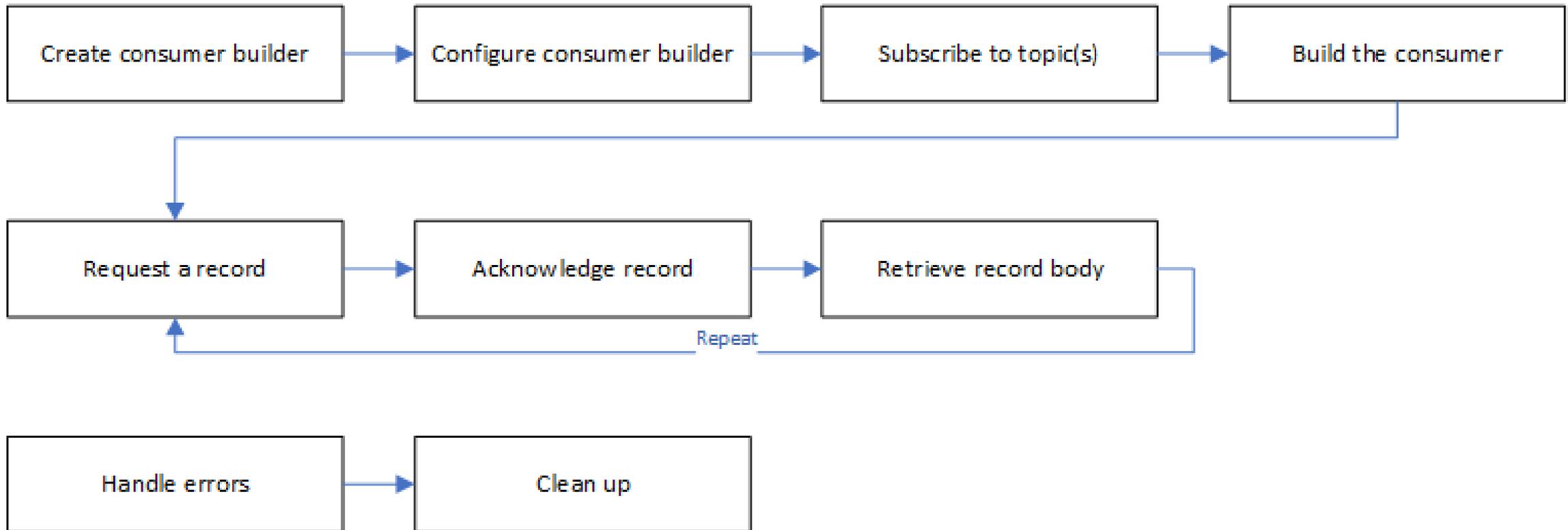




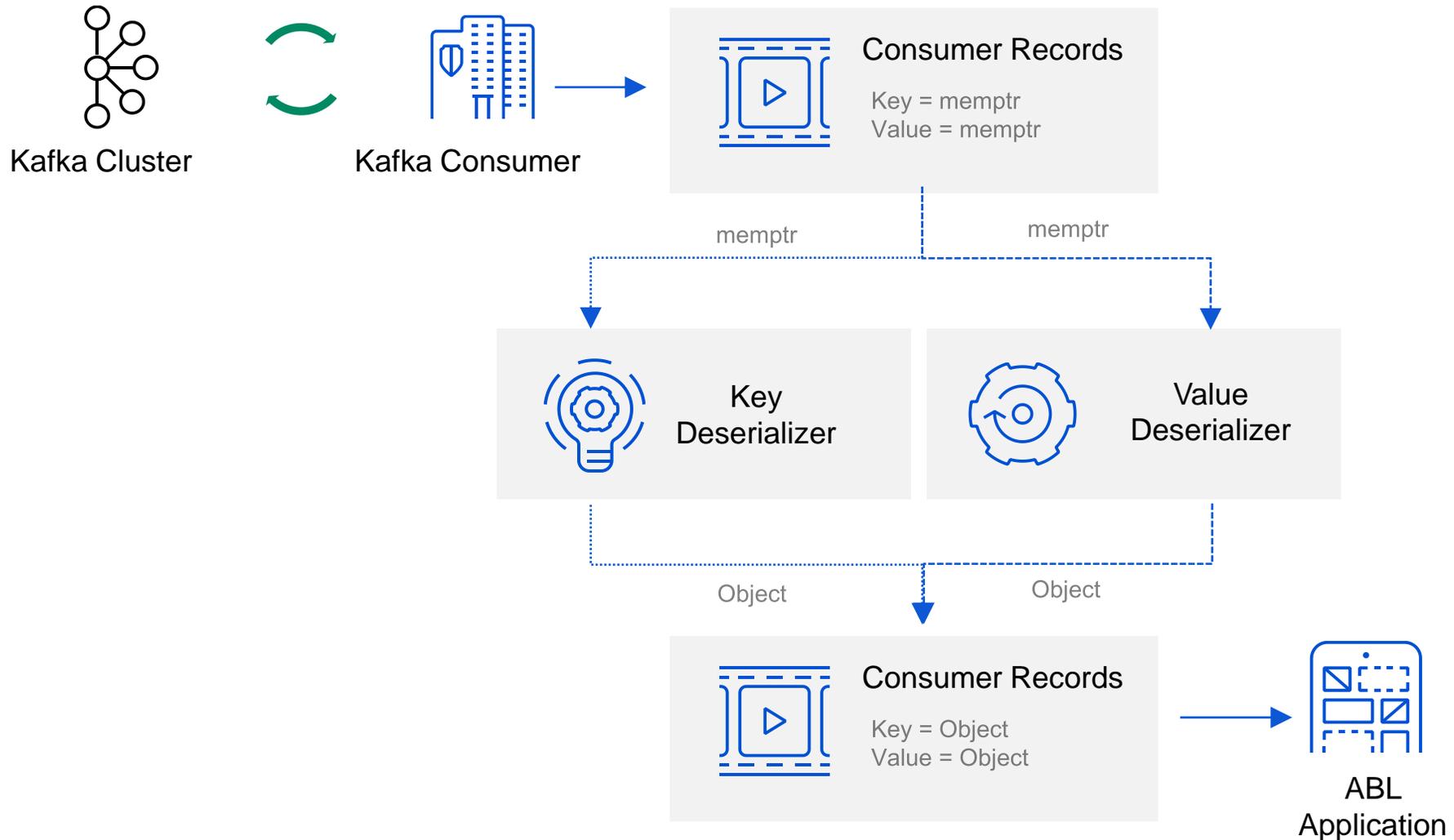
ABL Can Consume Kafka Messages

OpenEdge Consumer

- Reads and processes messages (records/events) from Kafka cluster
- Allowed in GUI^{***}, CHUI^{***}, batch. No PASOE. ^{***}no integ with event loop



Consumer Message Flow



DeSerializers

- Custom deserializers
- Provided deserializers

Serializer	Description
OpenEdge.Messaging.StringDeserializer	Simple string-base text. Converts bytes to strings and uses UTF-8 character encoding.
OpenEdge.Messaging.MemptrDeserializer	Pass-through serializer.
OpenEdge.Messaging.JsonDeserializer	Converts bytes to ABL JSON objects.

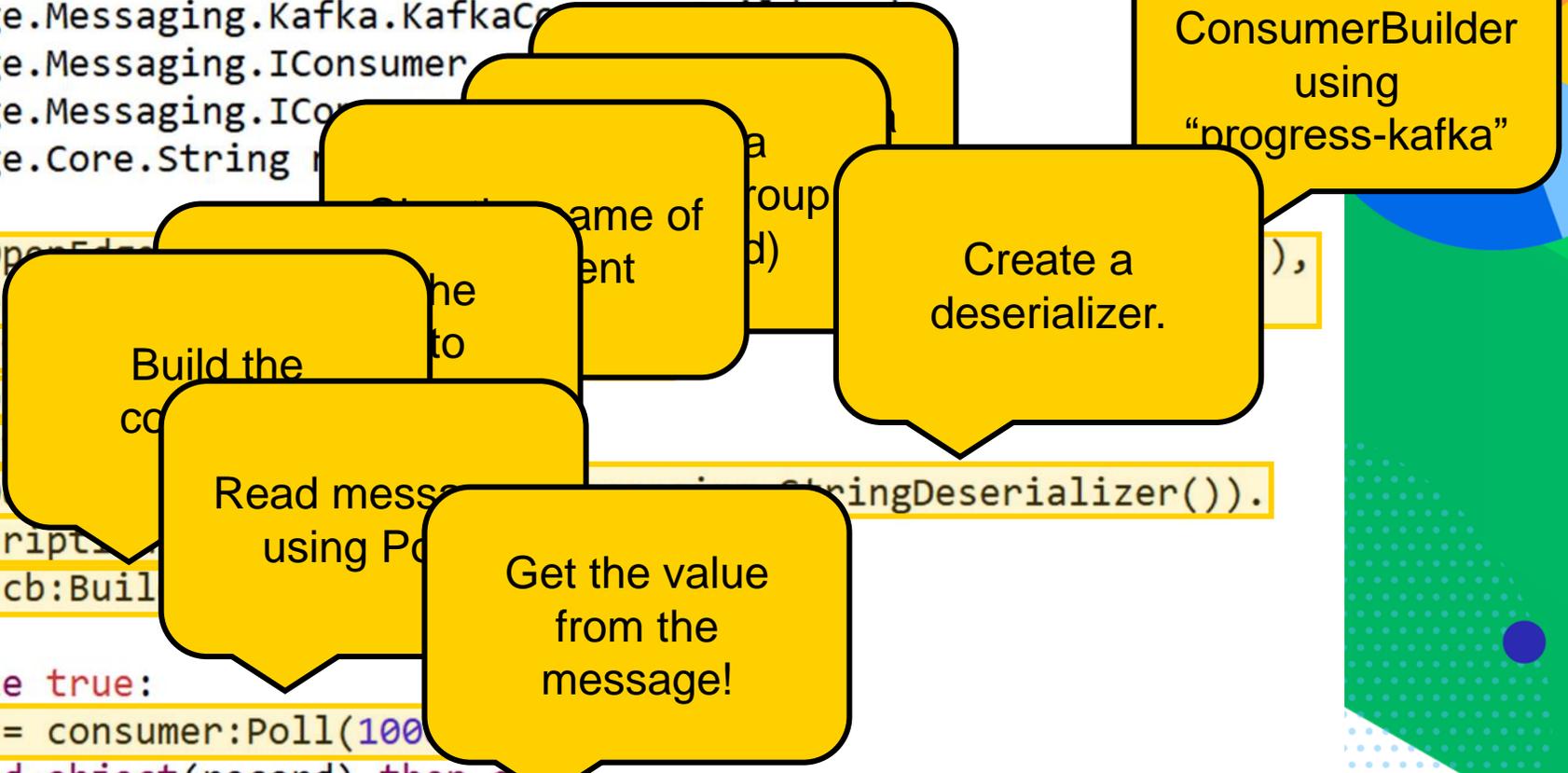
```
cb:SetBodyDeserializer( new StringDeserializer() ).  
cb:SetKeyDeserializer( new StringDeserializer() ).
```

Message Consumer

```
var OpenEdge.Messaging.Kafka.KafkaConsumer
var OpenEdge.Messaging.IConsumer
var OpenEdge.Messaging.IConsumer
var OpenEdge.Core.String
```

```
cb = cast(OpenEdge.Messaging.Kafka.KafkaConsumer,
OpenEdge.Messaging.IConsumer)
cb:SetBoots
cb:SetGroup
cb:SetClient
cb:SetBodyD
cb:AddSubscrip
consumer = cb:Build
```

```
repeat while true:
  record = consumer:Poll(100)
  if valid-object(record) then do
    messageBody = cast(record:Body, OpenEdge.Core.String).
    message string(messageBody:Value).
  end.
end.
```



Record Acknowledgement

- Tells the cluster where in the stream the consumer is
- On restart, the consumer can start where it left off
- By default, consumer automatically acknowledges
- Acknowledging can be done manually

```
cb:SetEnableAutoCommit( false ).
```

← Turn off auto commit

```
...
```

```
repeat while true:
```

```
    record = consumer:Poll( 1000 ).
```

```
    if valid-object( record ) then do:
```

```
        consumer:CommitOffset( record ).
```

← Tell broker you are finished with this message

```
        messageBody = cast( record:Body, JsonObject ).
```

```
        message messageBody:GetCharacter( "state" ).
```

```
    end.
```

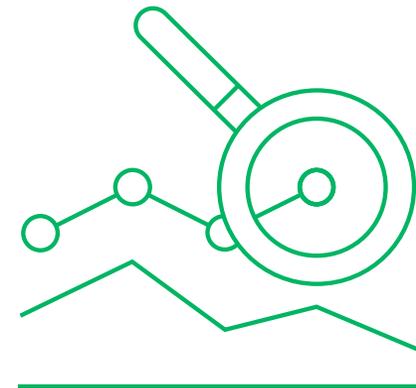
```
end.
```

DEMO



Recap

- Know a bit about Apache Kafka
 - Introduced to OpenEdge Messaging API
 - Kafka Producer in ABL
 - Kafka Consumer in ABL
-
- Capture data in real time from many event sources!



Customer Validation Program



Join the CVP

OpenEdge Customer Validation Program

- Actively influence the developer experience and future enhancements of Progress OpenEdge!
- Get Access to:

Roadmap surveys	Virtual open houses
Usability reviews	Quarterly objectives
Pre-release software	Sprint reviews
- <https://www.progress.com/openedge/customer-validation-program>

Where can I learn more?

Documentation

- OpenEdge content portal at <https://docs.progress.com>
- Progress® OpenEdge® ABL API Reference at <https://documentation.progress.com/output/oehttpclient>

Education

- Kafka <https://kafka.apache.org/>
- librdkafka <https://github.com/confluentinc/librdkafka>

Other

- <https://youtube.com> Search for: #kafka #confluent Tim Berglund
- <https://confluent.io>
- <https://github.com/confluentinc/librdkafka/blob/master/CONFIGURATION.md>

