Java Serialization for Big Data

Daniel Hinojosa

About Me..

Daniel Hinojosa
Programmer, Consultant, Trainer

On O'Reilly:
Testing in Scala (Book)
Beginning Scala Programming (Video)
Scala Beyond the Basics (Online Training)
TDD in Java (Online Training)
Getting Starting with Scala* (Online Training)
Scala The Next Level* (Online Training)

Contact:
dhinojosa@evolutionnext.com
@dhinojosa

https://github.com/dhinojosa/serialization-study

Rationale For Presentation

- Java Serialization is Slow
- Backwards Compatibility is an issue
- Many modern messaging and Big Data aggregation tools use these built in serialization tools
- Cross-language may also be an issue

Apache Avro

Apache Avro

- Created by Doug Cutting; Creator of Hadoop
- Serialization is defined by schema
- Schemas are JSON Based
- Codegen available at command line
- Codegen available by Maven Plugin
- Supports the following languages
  - C, C++, Java, Perl, Python, Ruby, PHP

Avro Name Origin

- Original British Aircraft Manufacturer WWI, WWII
- A.V. Roe and Company established 1910

AVRO AIRCRAFT

AVRO ™

Apache Avro Types

- Generic Records
  - Develop code that reflects your schema
- Reflection
  - Auto-create schema from an existing class
- Specific
  - Codegen your class from schema
  - This is the most common form

Avro Primitive Types

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

Avro Complex Types

```
{
  "namespace": "com.evolutionnext.avro",        Package
  "type": "record",
  "doc" : "An music album",                     Documentation
  "name": "Album",                              Name
  "fields": [                                   Fields
    {
      "name": "name",
      "type": "string"
    },
    {
      "name": "yearReleased",
      "type": [
        "int",
        "null"                                  Union Type
      ]
    }
  ]
}
```

Avro Field Options

Avro Field	Description
name	Name of the field
doc	Documentation
type	Type of the field
default	Default value
order	What order does this impact record?
aliases	Other names for the field

Avro Enum

```
{ "type" : "enum",
  "name" : "rainbowColors",
  "doc" : "Colors of the Rainbow",
  "symbols" : ["RED", "ORANGE", "YELLOW",
               "GREEN", "BLUE", "INDIGO",
               "VIOLET"]}
```

Avro Tools

```
java -jar avro-tools-1.8.2.jar compile schema <schema file> <destination>
```

avro-tools jar

your avsc file

destination

Apache Avro Demo

Google Protobuf

About Protobuf

- Developed at Google
- Full Name: Protocol Buffer
- Uses Proprietary Language
    - proto2
    - proto3
- Support for
    - C++
    - Java
    - Python
    - Go
    - Ruby
    - C#

Protobuf Installation

- Download Protocol Buffer
- Ensure that g++ Compiler Installed
- Extract Protocol Buffer
- Run the following:
  - `./configure`
  - `make`
  - `make check`
  - `sudo make install`
  - `protoc --version`

For Windows: https://github.com/google/protobuf/blob/master/src/README.md#c-installation---windows

Protobuf Simple Types

Proto Type	Java Type
double	double
float	float
sint32	int
sint64	long
bool	boolean
string	String
bytes	ByteString (protobuf)

Other unsigned types are available:
`https://developers.google.com/protocol-buffers/docs/proto`

Using Protoc

```
protoc -I=$SRC_DIR --java_out=$DST_DIR $SRC_DIR/yourproto.proto
```

Where to find imports

Where to output java code

Your definition

Protobuf Build Tool Plugins

- Maven: https://www.xolstice.org/protobuf-maven-plugin/
- Gradle: https://github.com/google/protobuf-gradle-plugin
- SBT: https://github.com/sbt/sbt-protobuf
- Leiningen: https://github.com/ninjudd/lein-protobuf

Google Protocol Buffers Demo

Colfer

Colfer

- https://github.com/pascaldekloe/colfer
- Fastest Serialization according to benchmarks
- Per documentation: “Suboptimal performance is treated like a bug.”
- Schema/Codegen Based
- Inspired by Google Protocol Buffers
- Codegen can be performed by command line or build tool plugin

Colfer

- Multiple Language Support
  - C, C++
  - Go
  - Java
  - JavaScript

Colfer Simple Types

Colfer Type	Java Type
double	double
float	float
int32	int
int64	long
bool	boolean
timestamp	java.time.Instant
text	String
binary	byte[]

Other unsigned types are available:
`https://github.com/pascaldekloe/colfer`

Colfer Complex Types

```
//Comments

package datebook

type appointment struct {
     id sint
     datetime timestamp
     duration int32
     asset asset
}

type asset struct {
     name string
     type string
}
```

Comments appear in generated code

Package

Complex Type

Reference to complex type

Colfer Compiler

- Using ***go*** run the following to obtain Colfer:
  - `go get -u github.com/pascaldekloe/colfer/cmd/colf`
- To Run:
  - **`colf -p com/xyzcorp Java template_dir`**

$$\underbrace{\texttt{com/xyzcorp}}_{\text{package}}\quad \underbrace{\texttt{Java}}_{\text{language}}\quad \underbrace{\texttt{template\_dir}}_{\text{files/directory of templates}}$$

Colfer Maven Plugin

- Due to relative obscurity of Colfer, there is only a maven plugin
- Perhaps for other plugins, configuring to execute the colf command line would be in order

Colfer Demo

Care with CodeGen

Many of the Codegen Polyglot Serializable solutions (Protobuf, have unsigned int, long but Java <u>does not</u> support unsigned values!

Care with Serializable

Most will not offer subclassing (which makes sense) and will likely be up to you to reconstitute your object hierarchy.

Kryo

Kryo

- https://github.com/EsotericSoftware/kryo
- Non-Schema, Full-Java Serialization Library
- Easy to Use

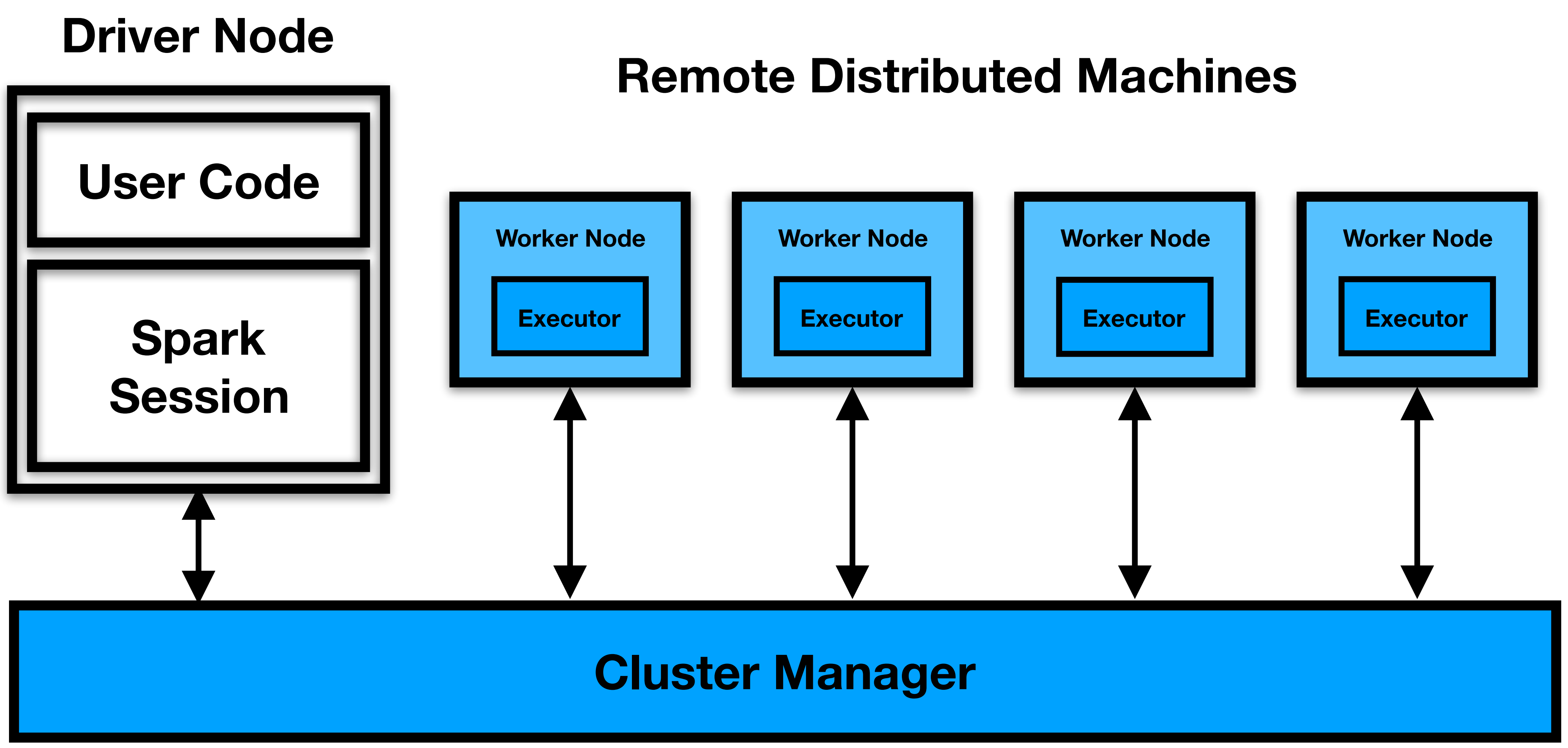
Kyro Demo

Benchmarks

https://github.com/eishay/jvm-serializers/wiki

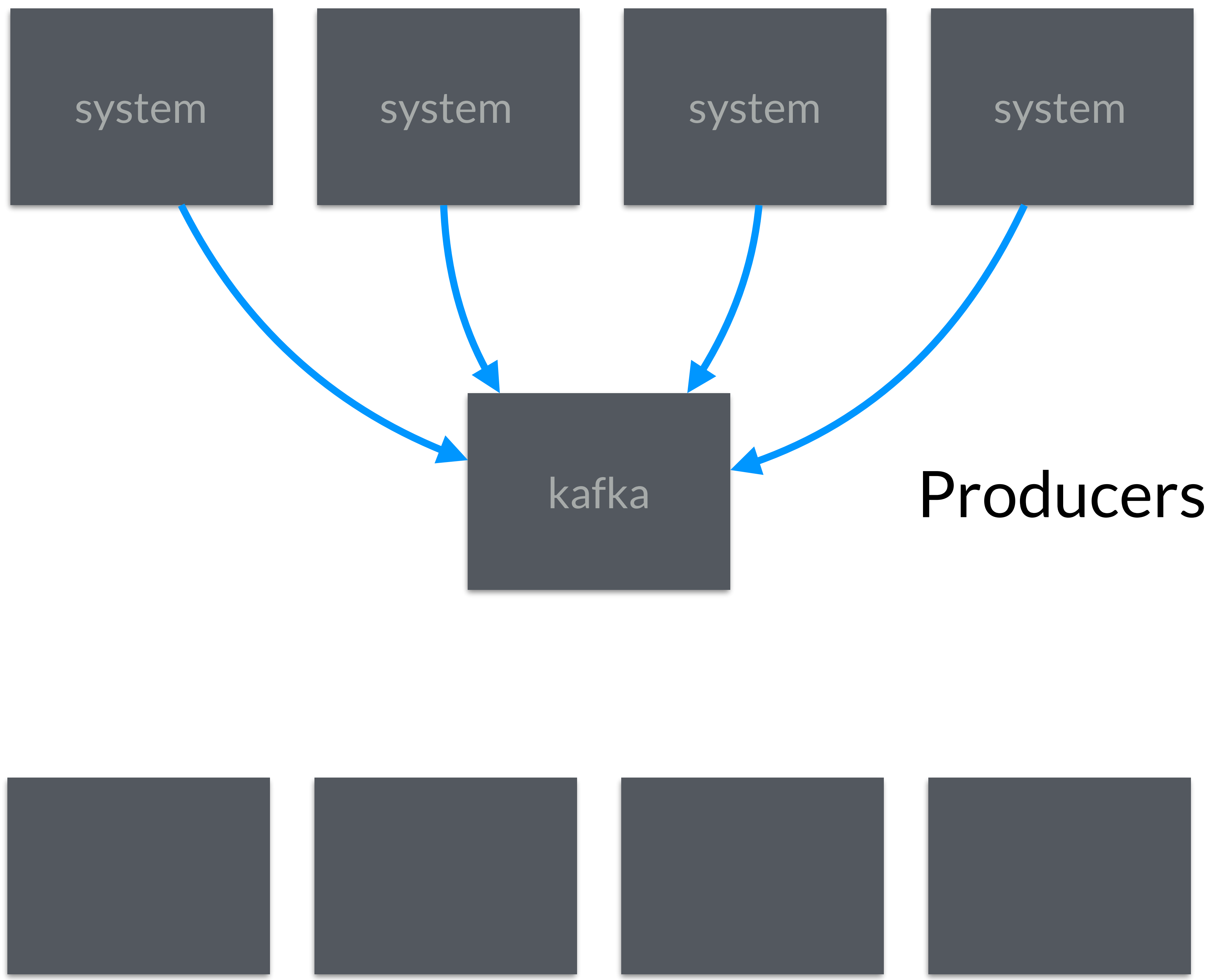
Kryo and Spark

How it works



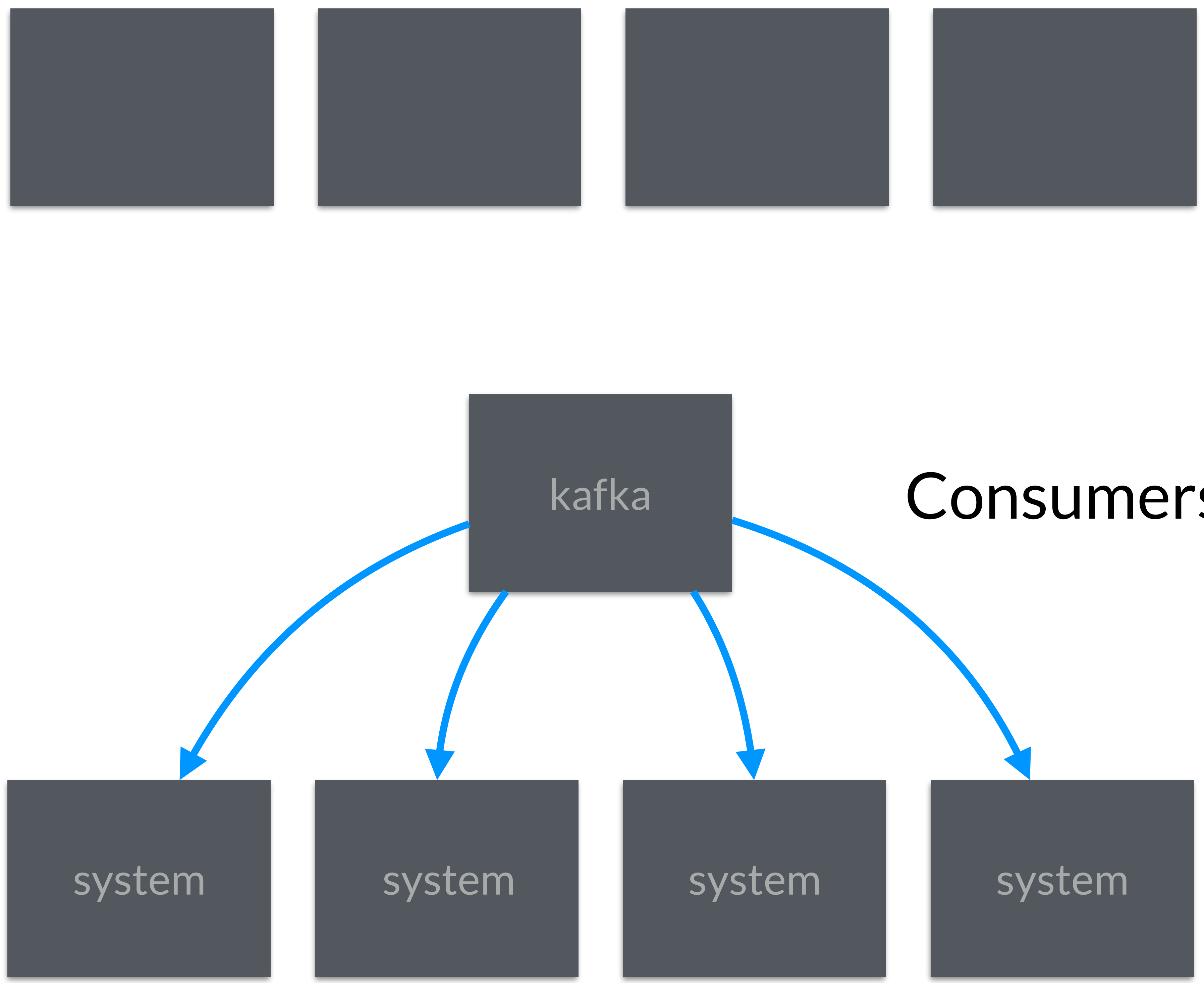
Driver Node
User Code
Spark Session
Remote Distributed Machines
Worker Node
Executor
Worker Node
Executor
Worker Node
Executor
Worker Node
Executor
Cluster Manager

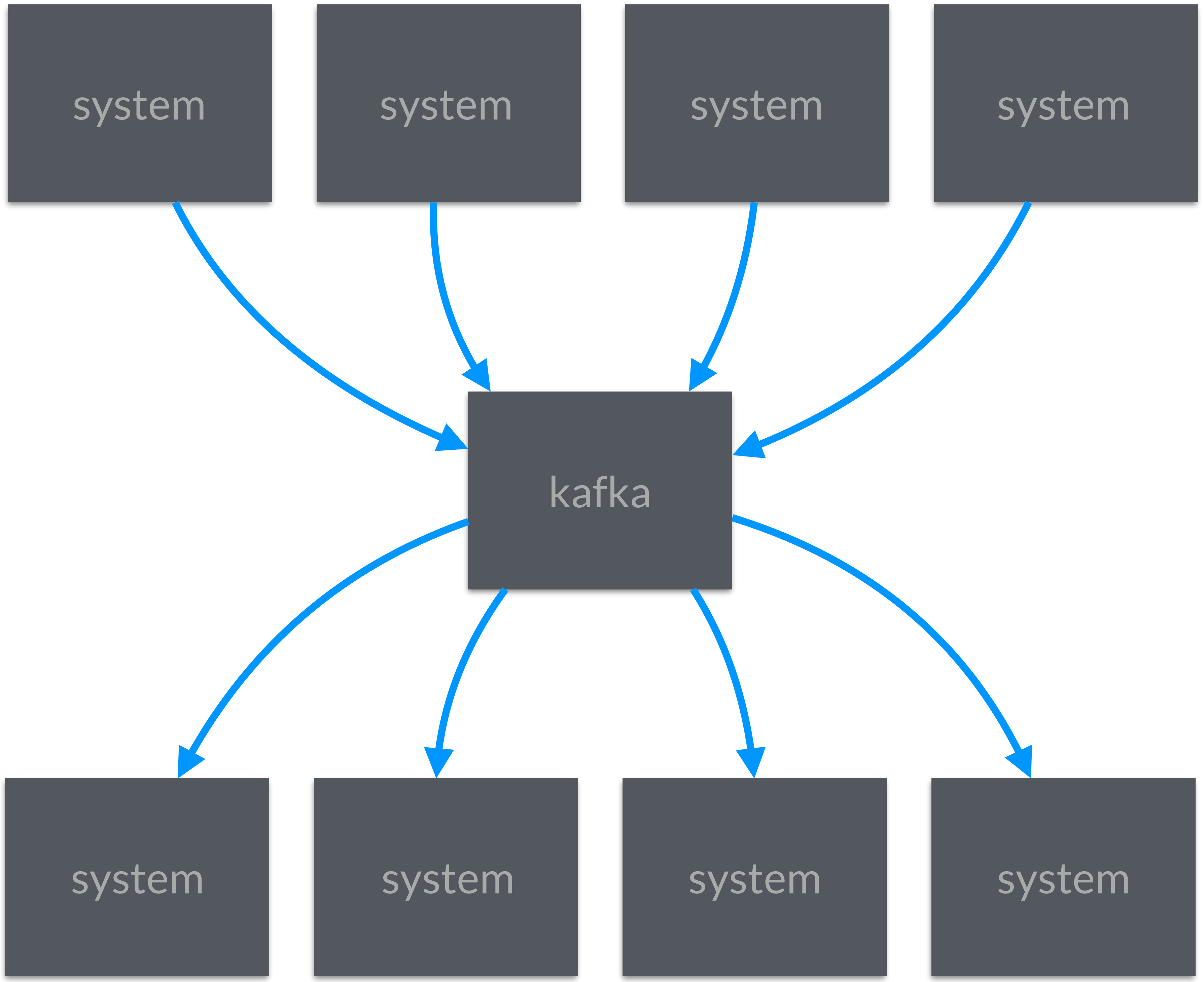
Avro and Kafka

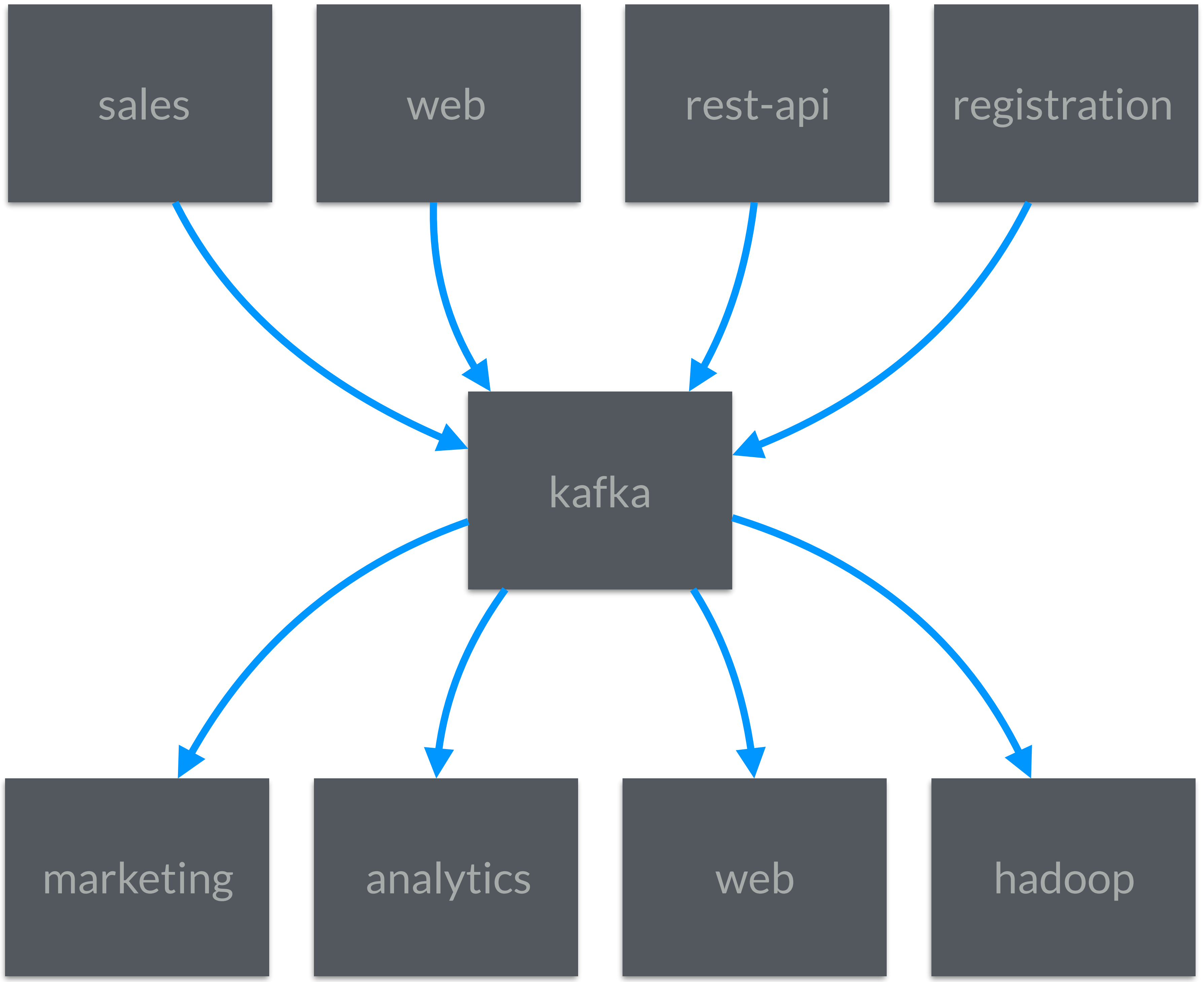


Producers

Consumers

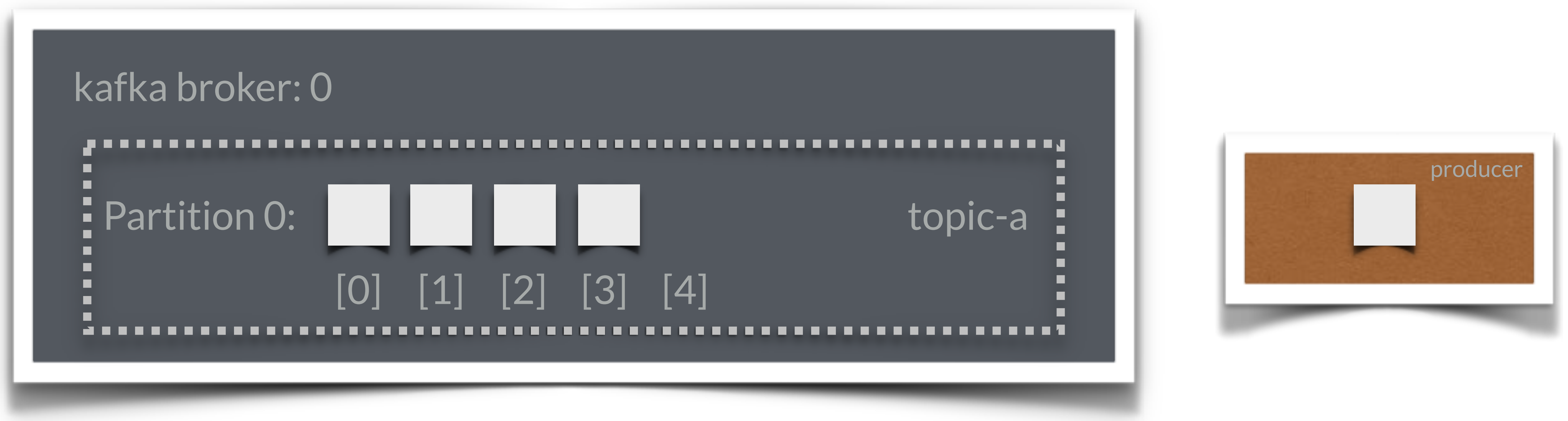


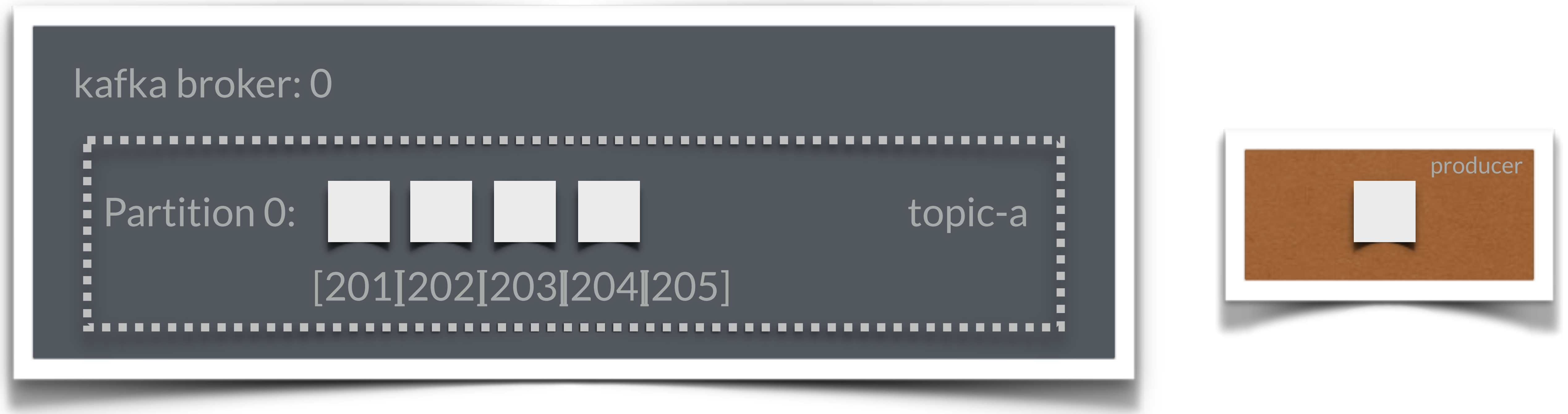

system
system
system
system
kafka
system
system
system
system



Oversimplied: A Producer can be a Consumer

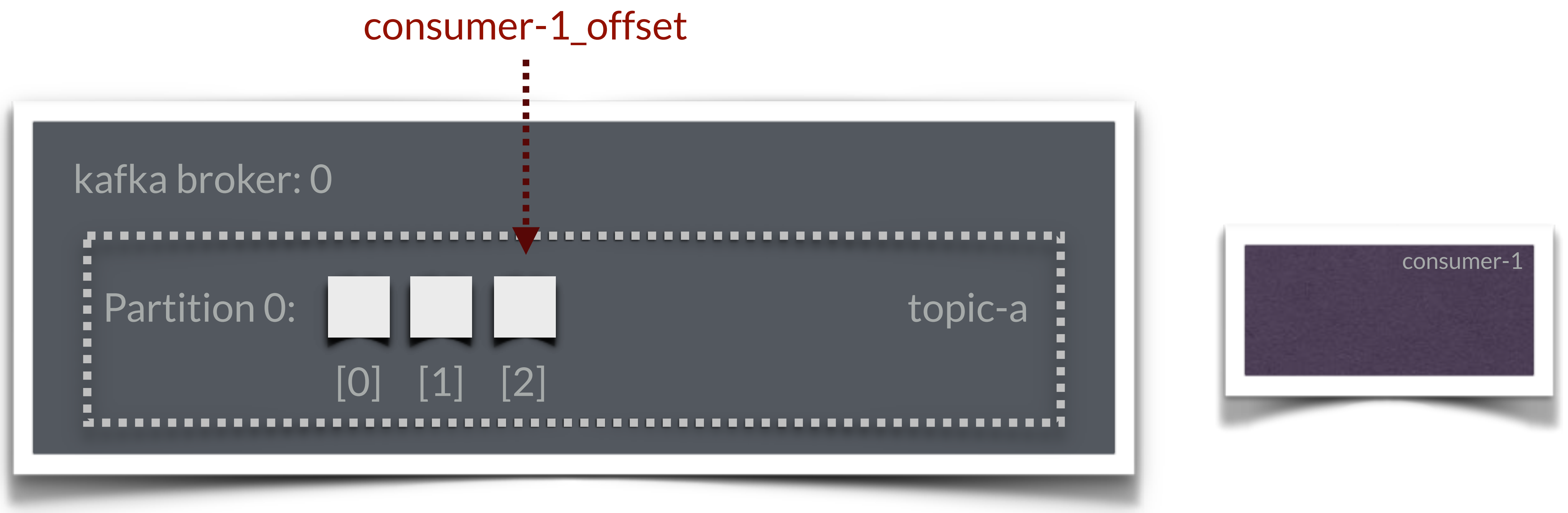
Kafka Producer

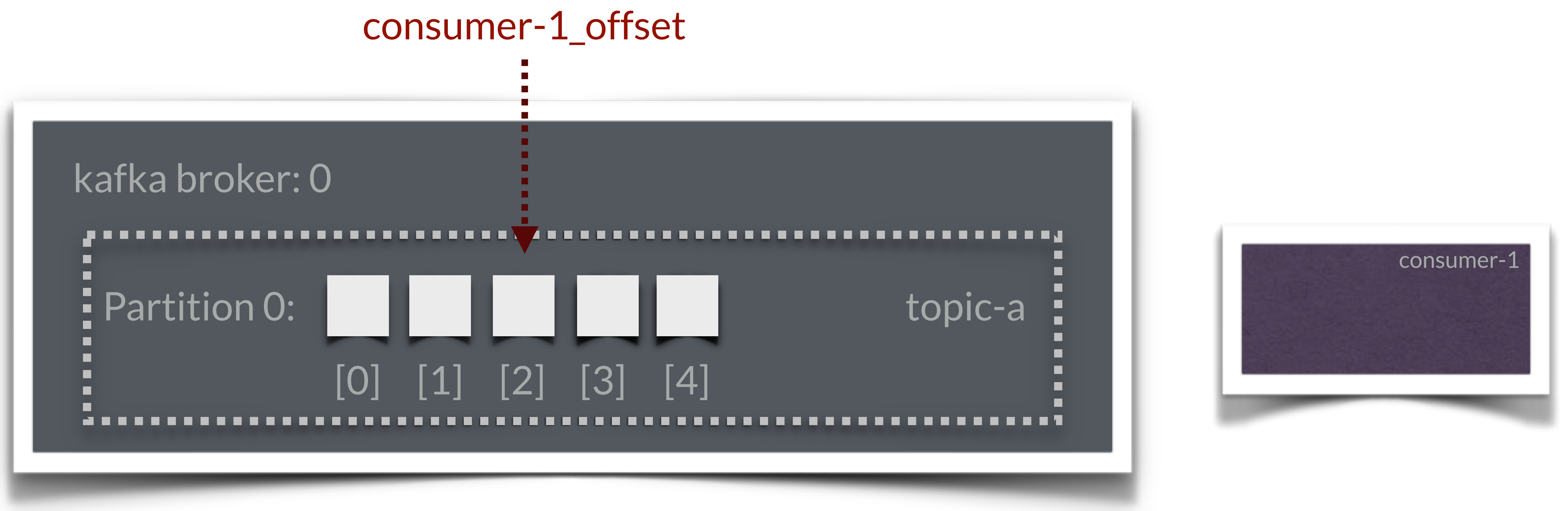

kafka broker: 0
Partition 0:
topic-a
[0]
[1]
[2]
[3]
[4]
producer

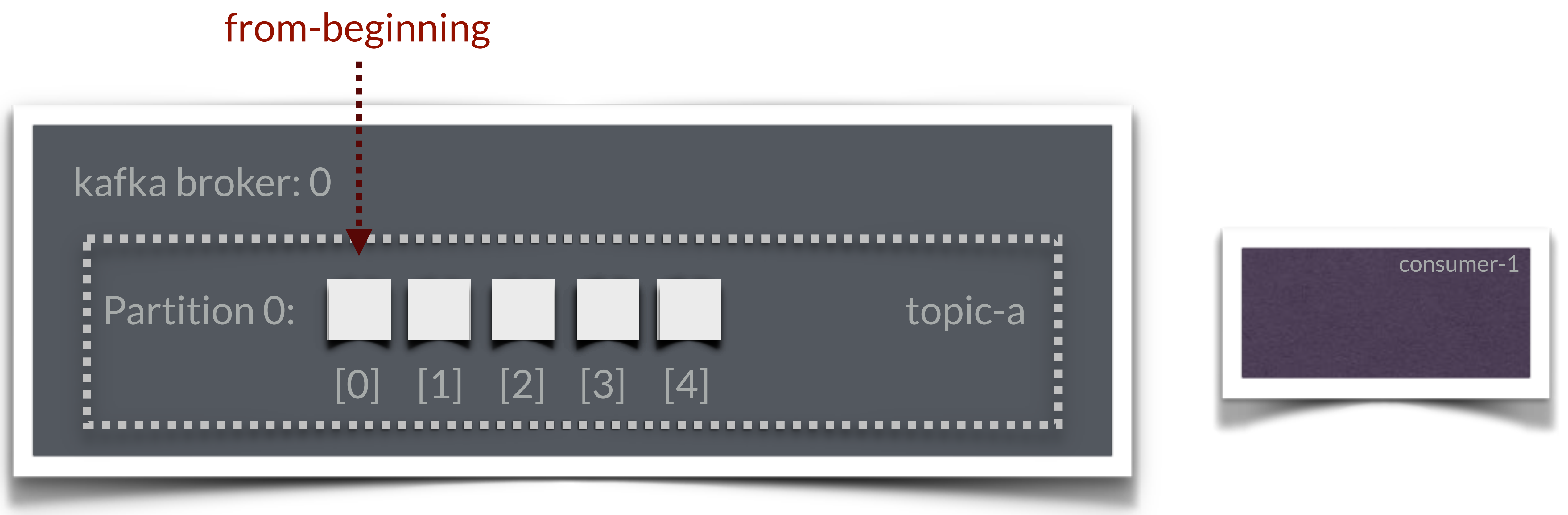


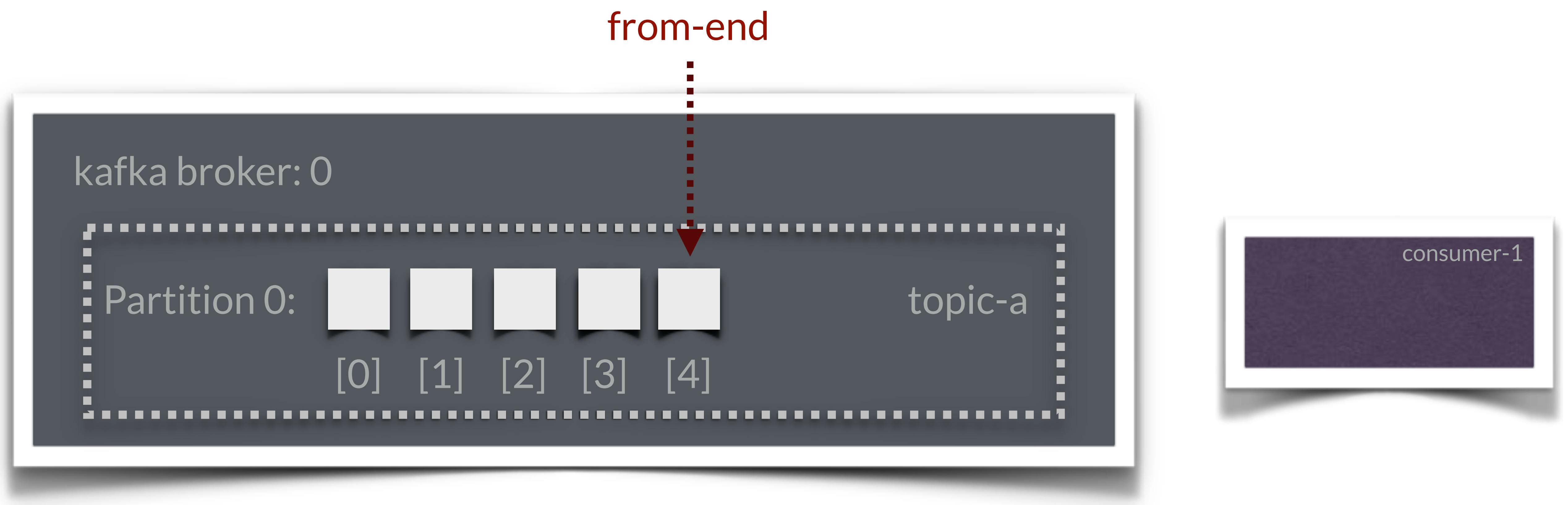
Retention: The **data is temporary**

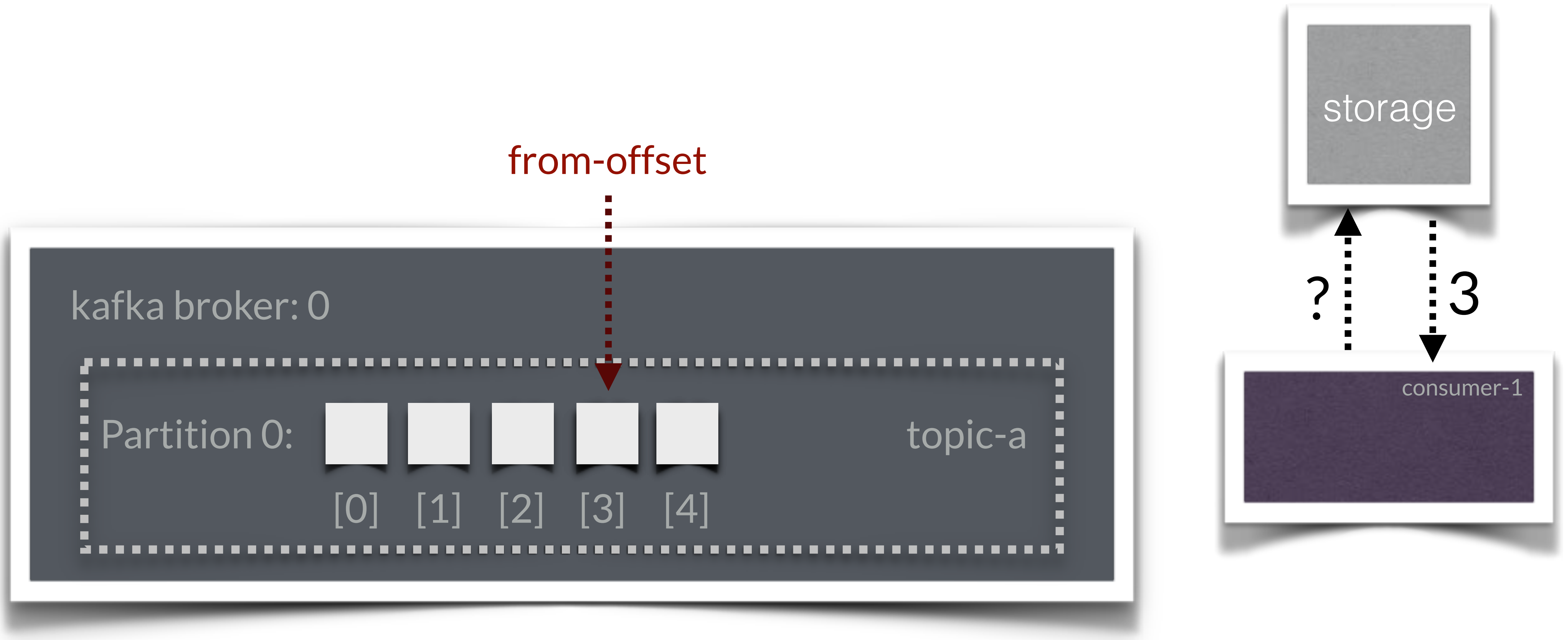
Kafka Consumer


consumer-1_offset
kafka broker: 0
Partition 0:
[0]
[1]
[2]
topic-a
consumer-1


consumer-1_offset
kafka broker: 0
Partition 0:
[0]
[1]
[2]
[3]
[4]
topic-a
consumer-1

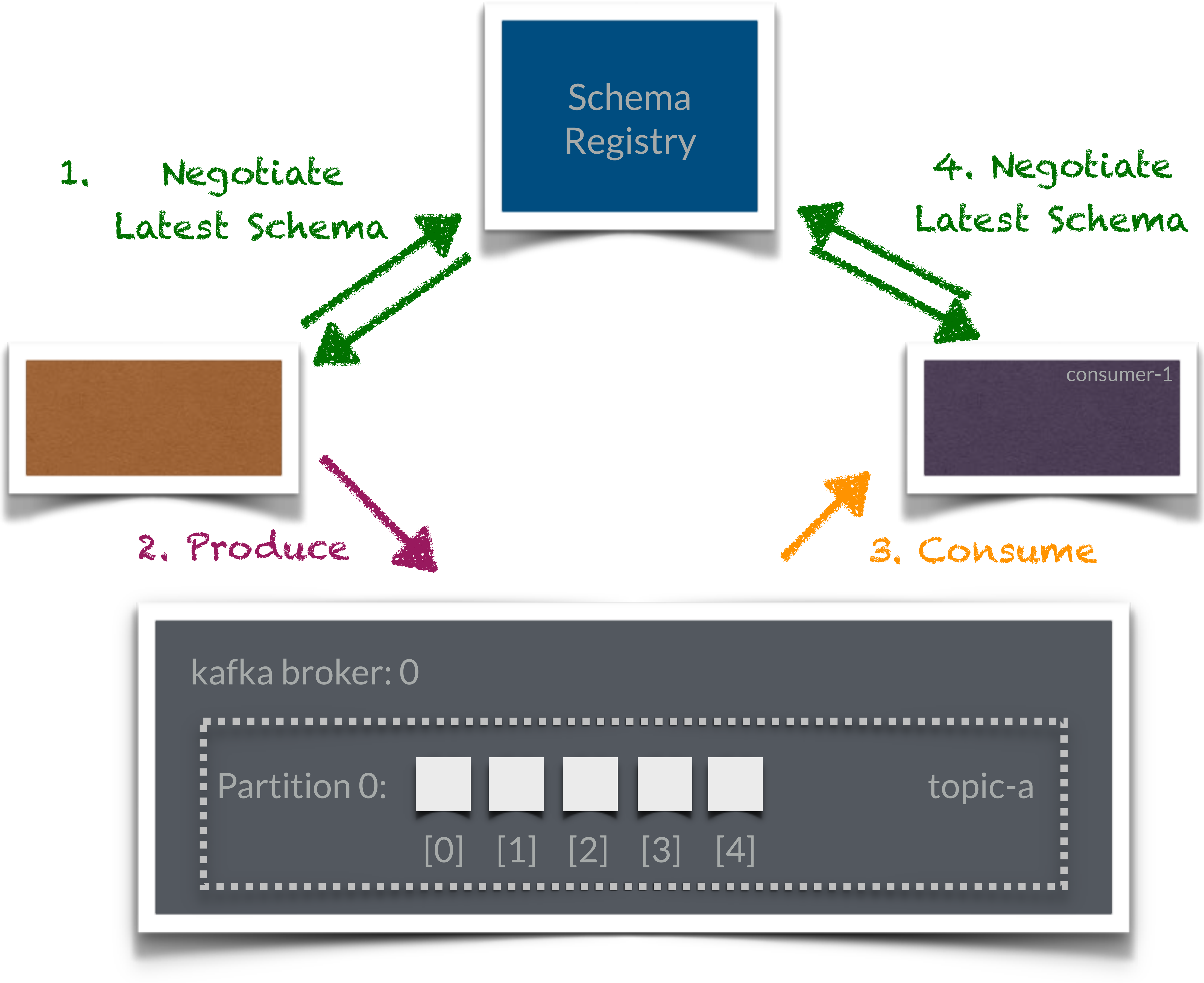

from-beginning
kafka broker: 0
Partition 0:
[0]
[1]
[2]
[3]
[4]
topic-a
consumer-1


from-end
kafka broker: 0
Partition 0:
[0]
[1]
[2]
[3]
[4]
topic-a
consumer-1


from-offset
kafka broker: 0
Partition 0:
[0]
[1]
[2]
[3]
[4]
topic-a
storage
?
3
consumer-1

Kafka Serialization Demo

Confluent Platform


Schema Registry
1. Negotiate Latest Schema
4. Negotiate Latest Schema
consumer-1
2. Produce
3. Consume
kafka broker: 0
Partition 0:
[0] [1] [2] [3] [4]
topic-a

Thank You

Email: dhinojosa@evolutionext.com
Twitter: @dhinojosa