Apache Kafka, a distributed persistent transactional log

Ugo Landini - Staff Solutions Engineer

Last updated: 28/06/23

> whoami

apiVersion: **confluent**/v1
kind: staff engineer
metadata:
  name: ugo landini
  nick: **ugol**
  email: ugo@confluent.io, ugo.landini@gmail.com
  namespace: **confluent**
  annotations: apache/committer, oss lover, distributed geek
  site: https://ugol.io
  labels:
    family: dad of two
    prev_companies: sun microsystems, vmware, red hat
  spec:
    replicas: 1
    containers:
      - image: github.com/ugol:latest

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

First look at Kafka
(from a cloud perspective)

2048

SCORE 1512 | BEST 6056

Scoreboard | New Game

2	4	4	2
32	16	4	2
64	32	8	4
4	128	32	8

HOW TO PLAY: Use your **arrow keys** to move the tiles. When two tiles with the same number touch, they **merge into one!**

NOTE: This game is the powered by Confluent Cloud. You can recreate this demo following self-paced workshop.

Demo by Gianluca Natali. Based on 2048 by Gabriele Cirulli.

https://gianlucanatali.github.io/streaming-games/index.html

Copyright 2020, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

ksqlDB

Schema Registry

Connect

Fully managed Apache Kafka

aws

Google Cloud

Azure

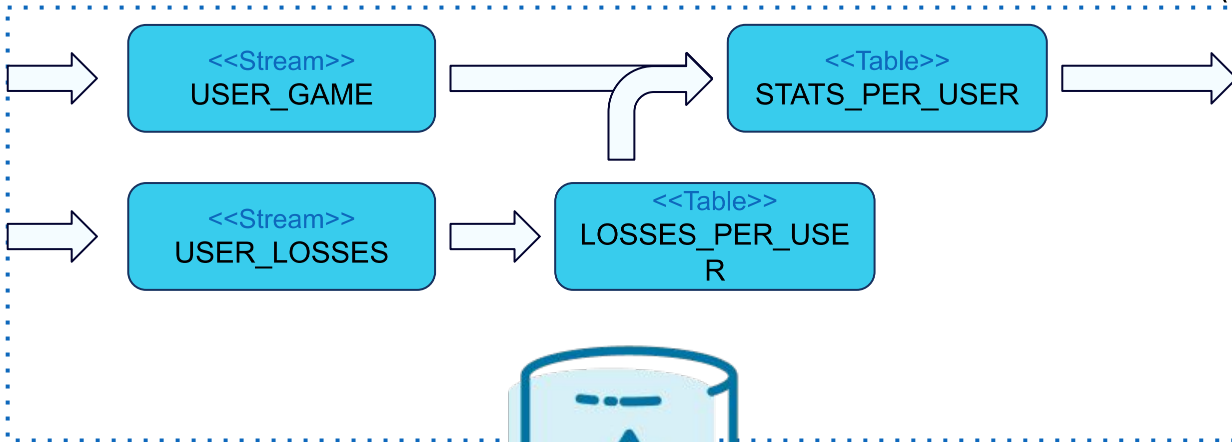
Connect

Copyright 2020, Confluent, Inc. All rights reserved. This document

INSERT
INTO...

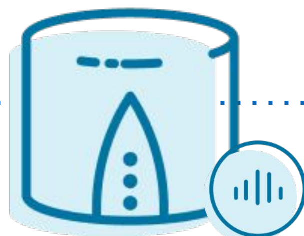
CREATE TABLE
FROM SELECT ..
JOIN...
GROUP BY ...

SELECT USER, HIGHEST_SCORE,
HIGHEST_LEVEL,
TOTAL_LOSSES from
STATS_PER_USER
WHERE USER IN (...)



PAC-MAN — BACK TO GAME —

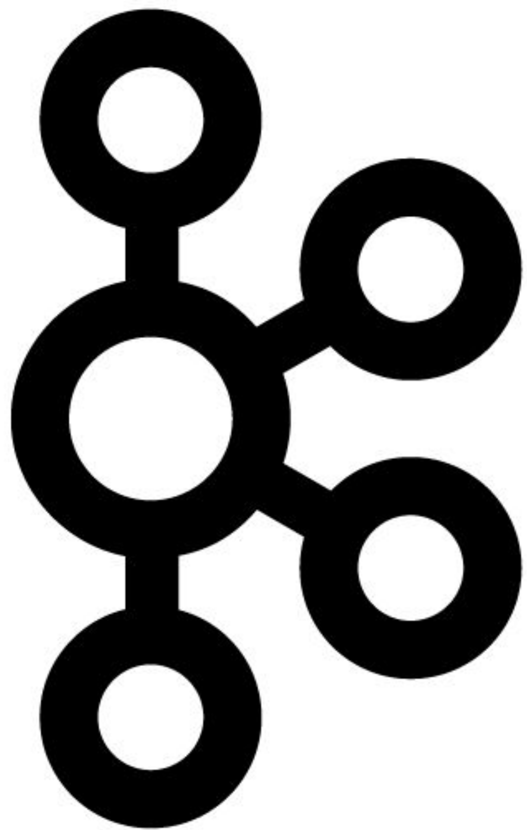
RANK	NAME	SCORE	LEVEL	LOSSES
*1	Gianluca Maestro Natali	213760	25	0
*2	Pac Master	200040	17	0
*3	ugol	67800	7	1
*4	Jordan	65120	8	0
*5	JT	60900	9	0
*6	MJ	15860	3	0
*7	xPer	14640	3	1
*8	UJ Salvi	14530	3	0
*9	Chris	13040	3	4



ksqlDB in Confluent Cloud

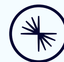
Copyright 2020, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

Kafka 101



APACHE
kafka

Some Kafka concepts to grasp



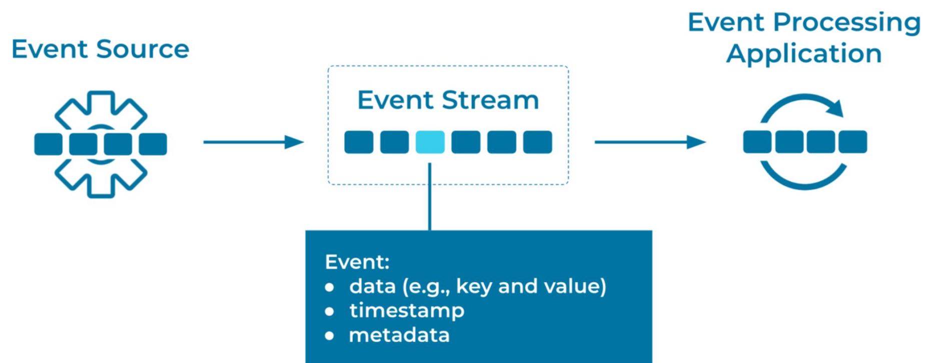
- **Events**
- **Topics**
  - Partitions
  - Replica
- **Producers**
  - Acks
  - ISR
- **Consumer**
  - Consumer Groups
  -

confluent kafka topic **produce** test --parse-key --delimiter ,
confluent kafka topic **consume** test --from-beginning

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

Events

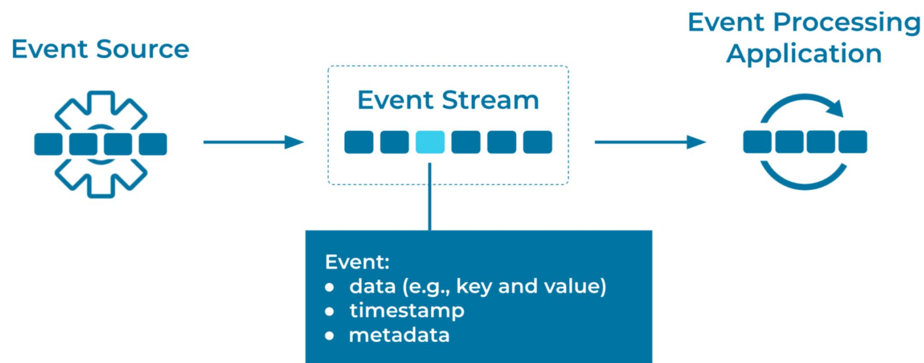
An event represents an immutable fact about something that happened



Events

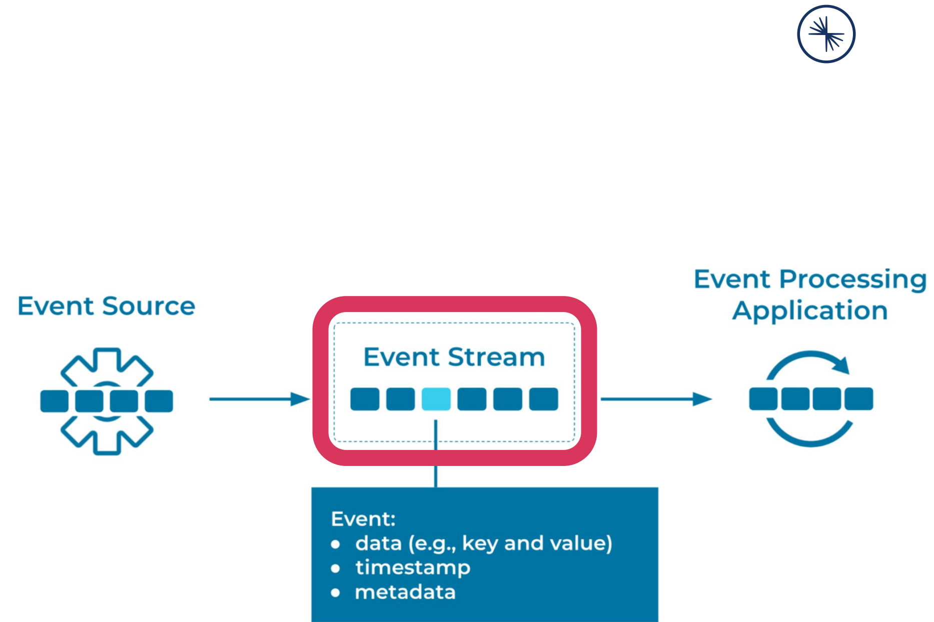
An event represents an immutable fact about something that happened

- Examples of events are customer orders, payments, activities, or measurements



Event Streams

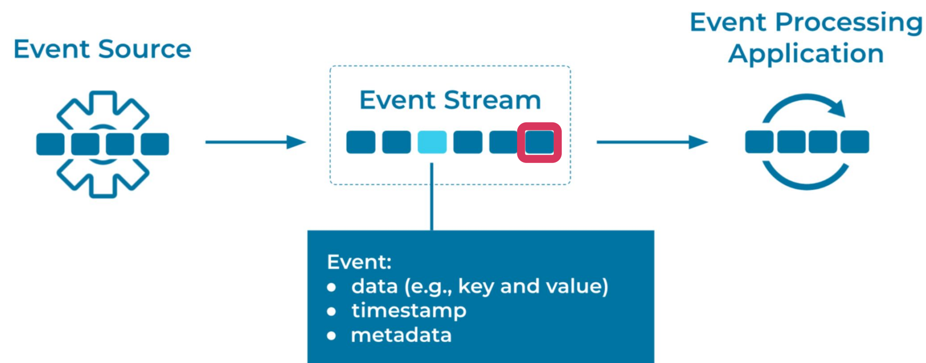
Events are produced to, stored in, and consumed from an event stream



Event Streams

Events are produced to, stored in, and consumed from an event stream

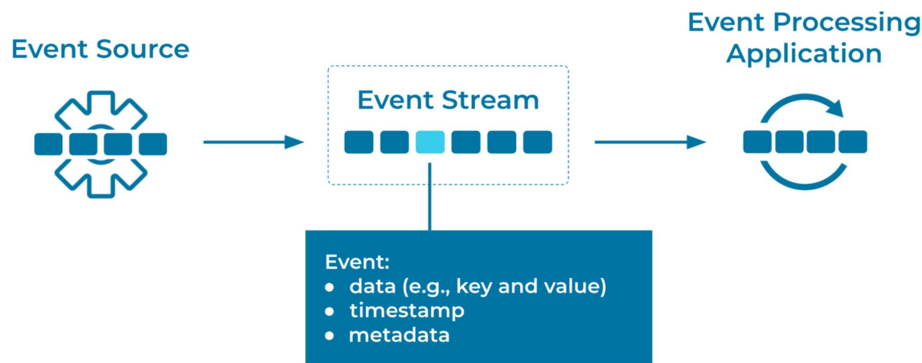
- New events are always appended to the end of the event stream



Event Streams

Events are produced to, stored in, and consumed from an event stream

- New events are always appended to the end of the event stream
  - Events are delivered to consumers in this append order

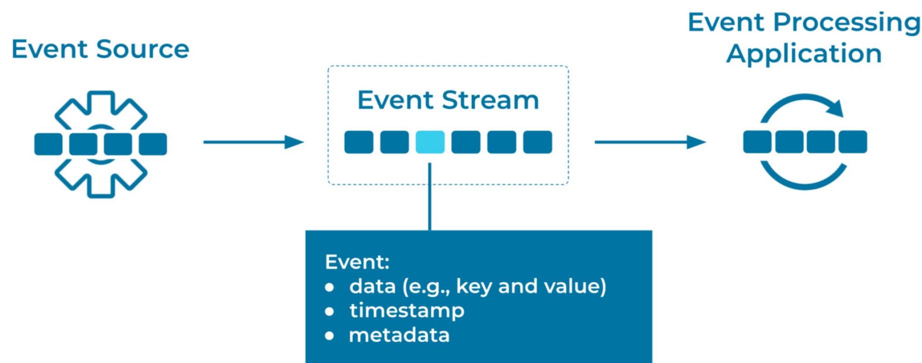


Event Streams

Events are produced to, stored in, and consumed from an event stream

- New events are always appended to the end of the event stream
  - Events are delivered to consumers in this append order

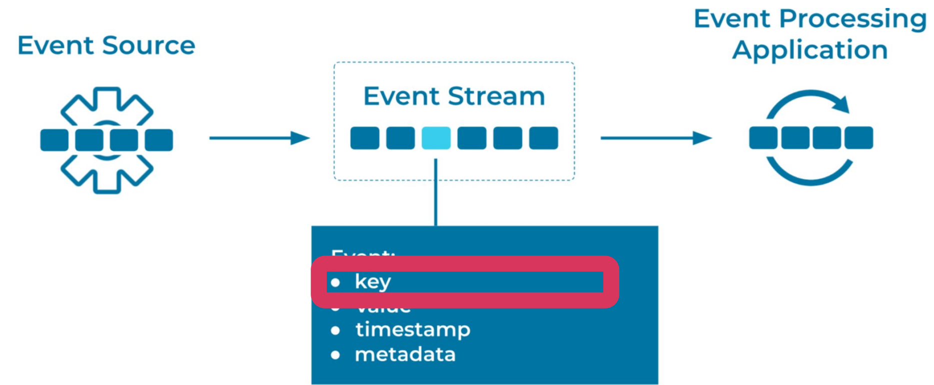
Once events have been written, they are immutable



Kafka Events

Kafka events contain:

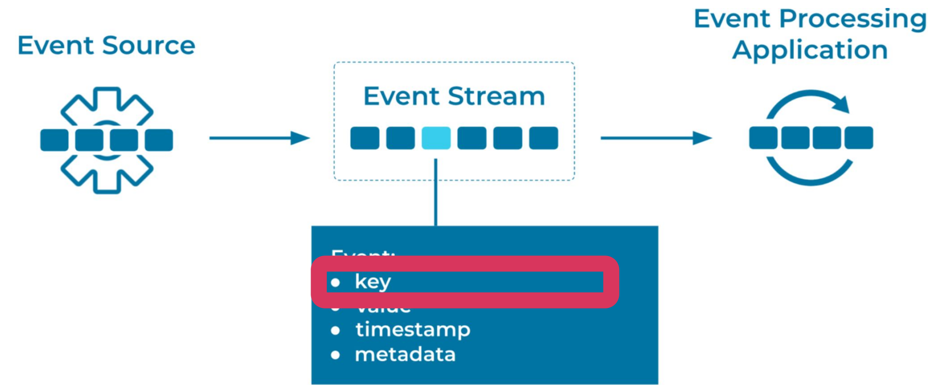
- Key: identifies events related to a specific entity



Kafka Events

Kafka events contain:

- Key: identifies events related to a specific entity

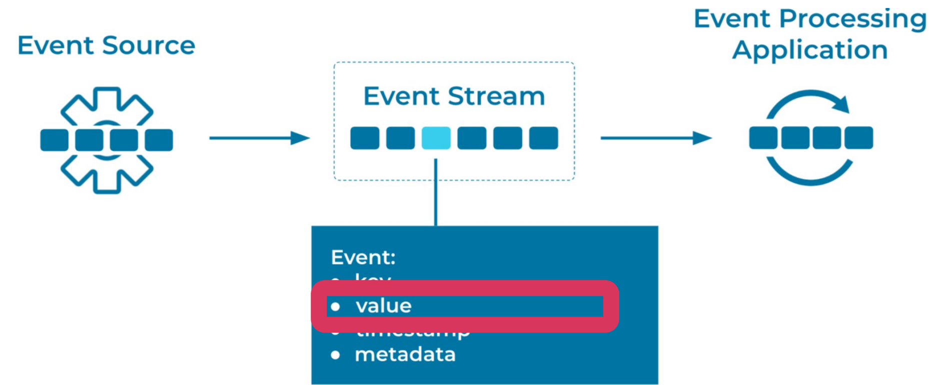



```
{
  "key": 2,
  "value": {
    "user_id": 2,
    "username": "user1234",
    "email": "user1234@mail.com",
    "level": "GOLD"
  }
}
```

Kafka Events

Kafka events contain:

- Key: identifies events related to a specific entity
- Value: data that describes the event

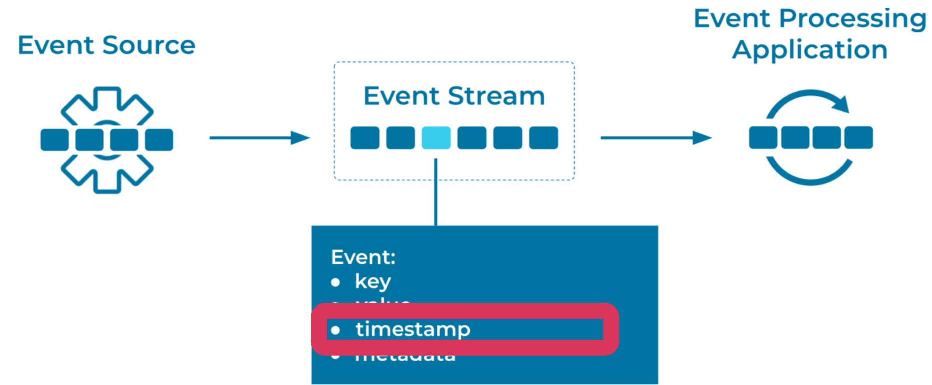



```
{
  "key": 2,
  "value": {
    "user_id": 2,
    "username": "user1234",
    "email": "user1234@mail.com",
    "level": "GOLD"
  }
}
```

Kafka Events

Kafka events contain:

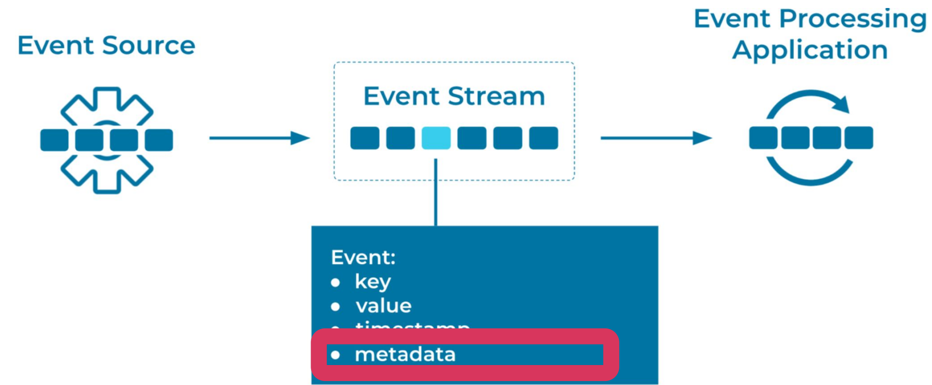
- Key: identifies events related to a specific entity
- Value: data that describes the event
- Timestamp: denotes when the event was created



Kafka Events

Kafka events contain:

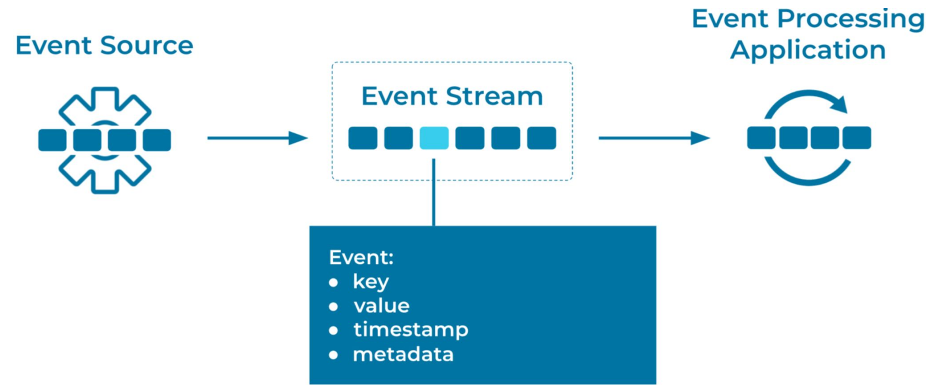
- Key: identifies events related to a specific entity
- Value: data that describes the event
- Timestamp: denotes when the event was created
- Metadata: optional



Kafka Events

Kafka events are also referred to as "records" and "messages"

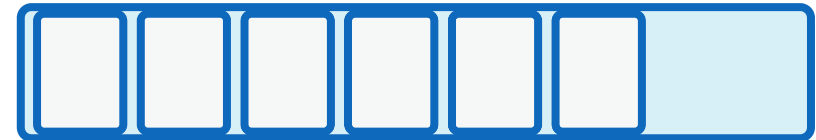
- event = record = message



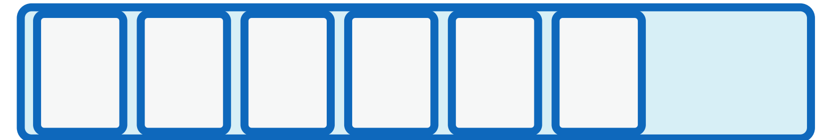
Kafka Topics

Named container of “related” events

- Example: a topic that stores all customer orders



account-deposits

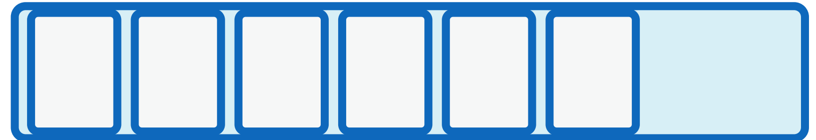


account-balance

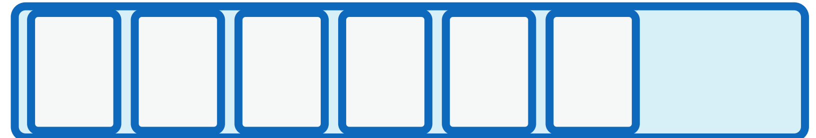
Kafka Topics

Named container of “related” events

- Example: a topic that stores all customer orders
- Kafka clusters typically contain many topics



account-deposits



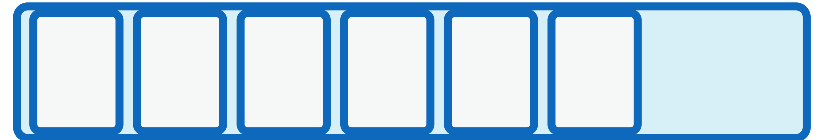
account-balance

Kafka Topics

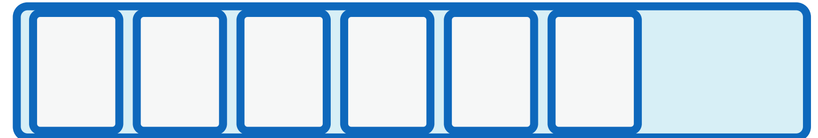
Named container of “related” events

- Example: a topic that stores all customer orders
- Kafka clusters typically contain many topics
  - Consumers subscribe at the topic level

Event Stream



account-deposits



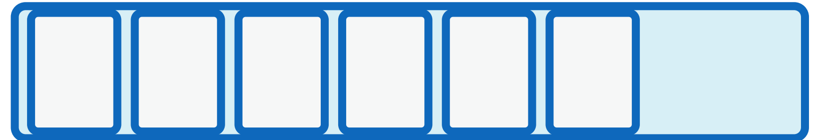
account-balance

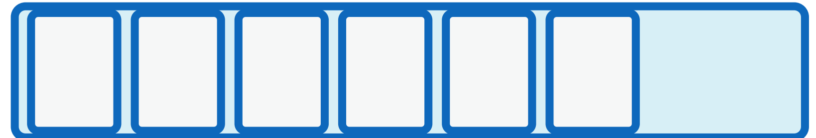
Kafka Topics

Named container of “related” events

- Example: a topic that stores all customer orders
- Kafka clusters typically contain many topics
  - Consumers subscribe at the topic level

Take the form of a durable log (data structure) of events





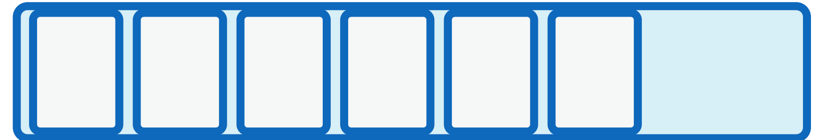
Kafka Topics

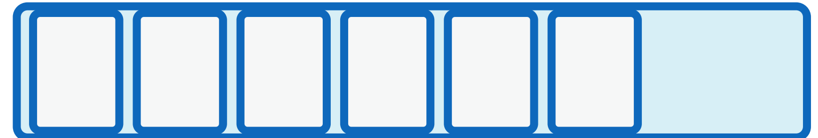
Named container of “related” events

- Example: a topic that stores all customer orders
- Kafka clusters typically contain many topics
  - Consumers subscribe at the topic level

Take the form of a durable log (data structure) of events

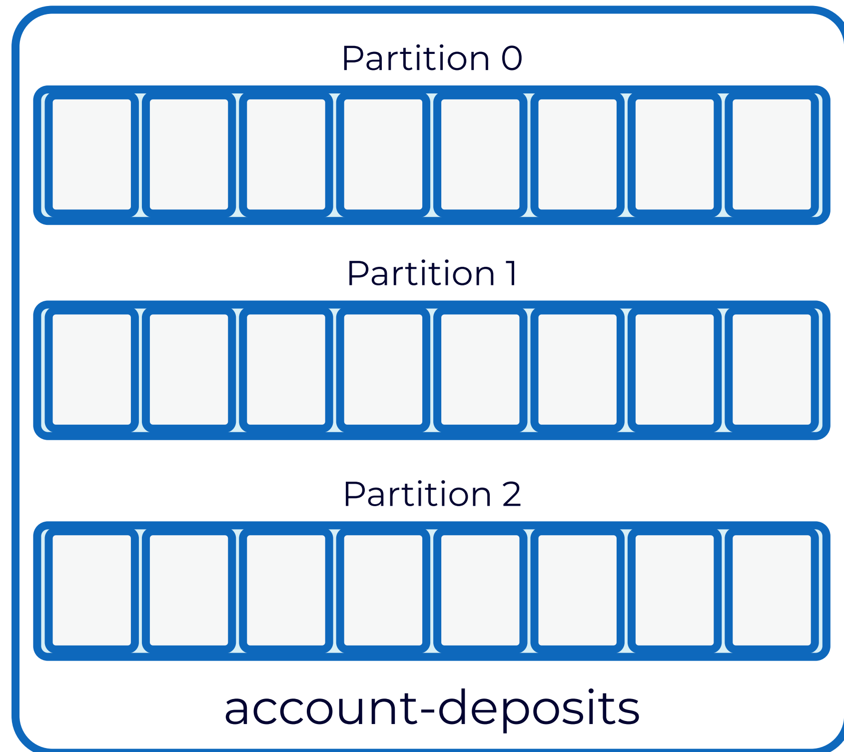
- Data retention period is configurable





Topic Partitions

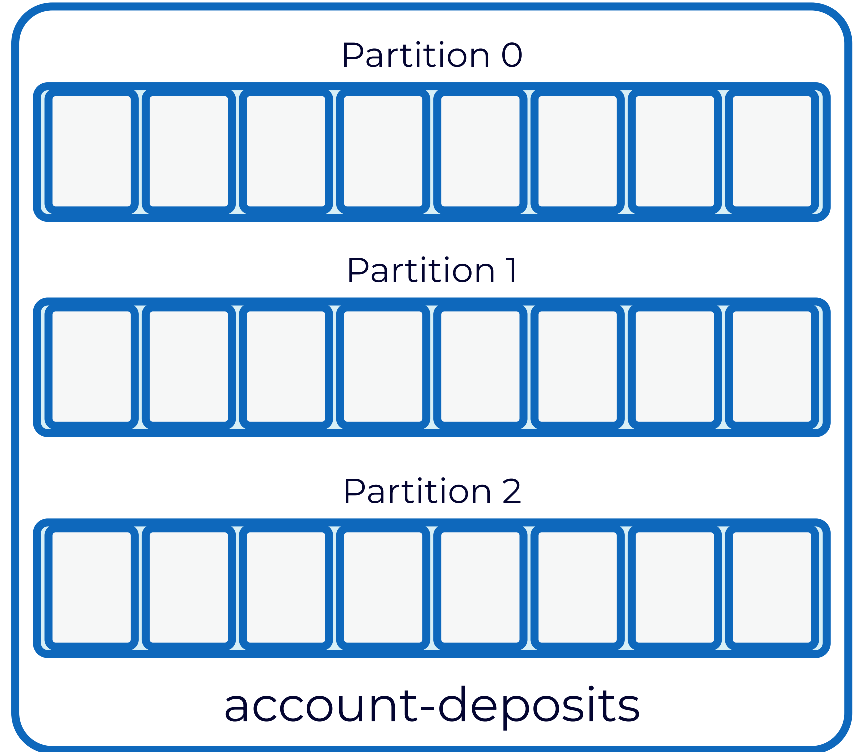
A topic consists of partitions



Topic Partitions

A topic consists of partitions

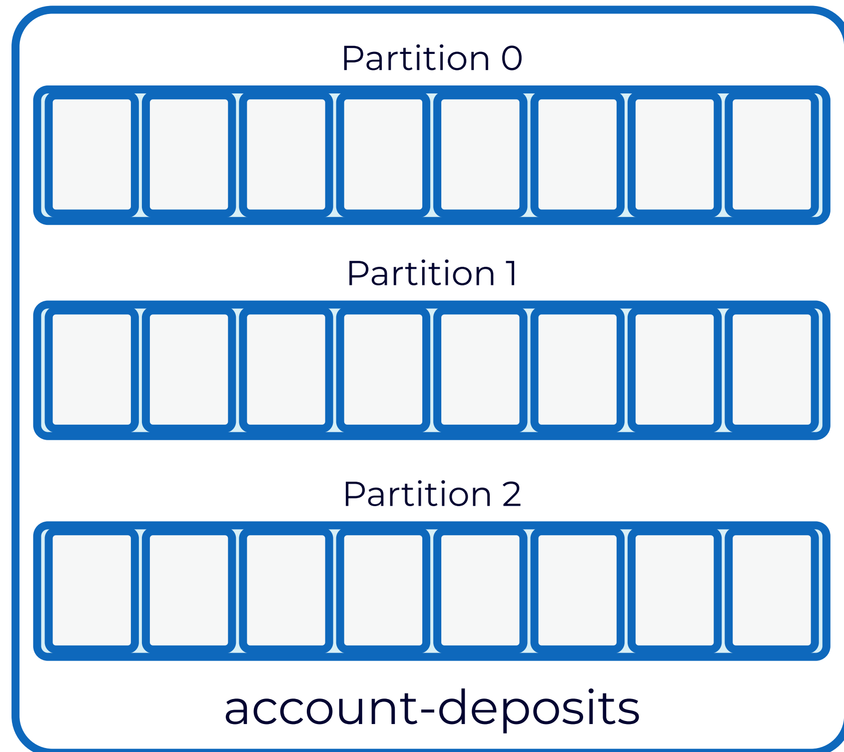
- Partitions provide scalability



Topic Partitions

A topic consists of partitions

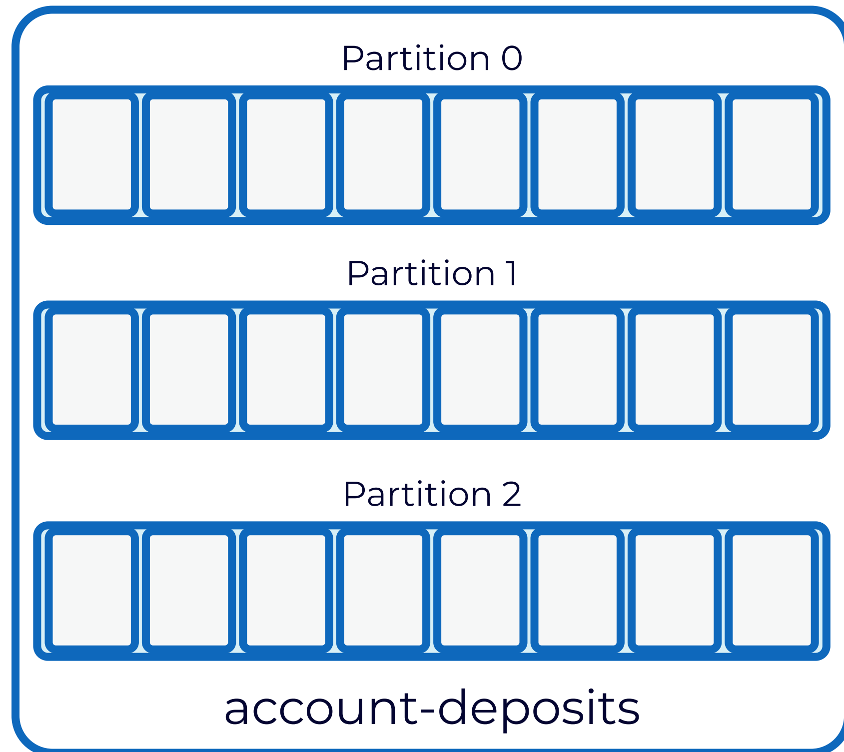
- Partitions provide scalability
- Partitions are evenly distributed across brokers within the Kafka cluster



Topic Partitions

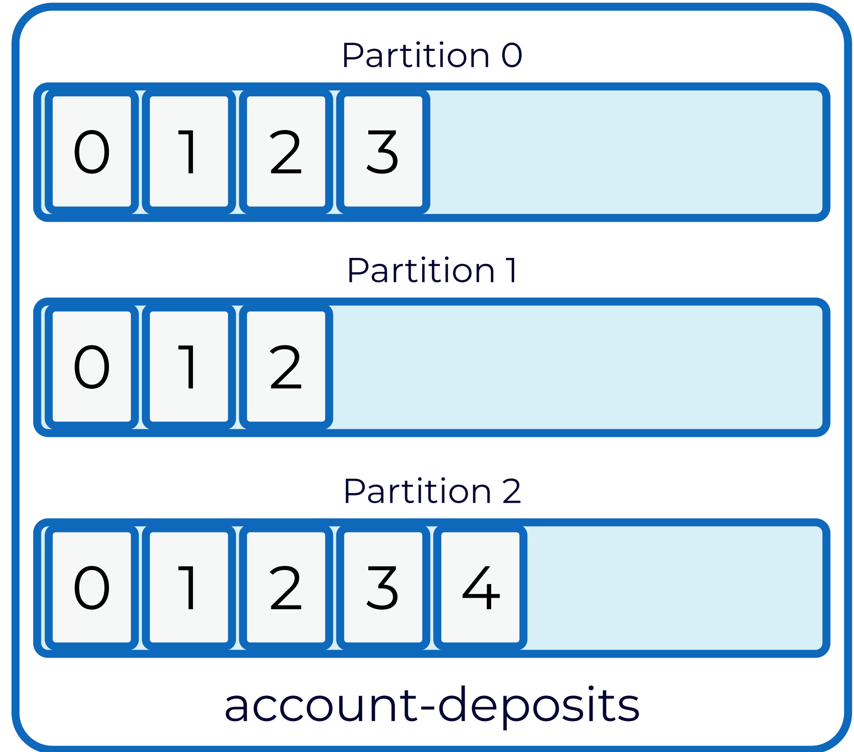
A topic consists of partitions

- Partitions provide scalability
- Partitions are evenly distributed across brokers within the Kafka cluster
  - With Confluent Tiered Storage, partitions can be split between brokers and object storage



Partition Offsets

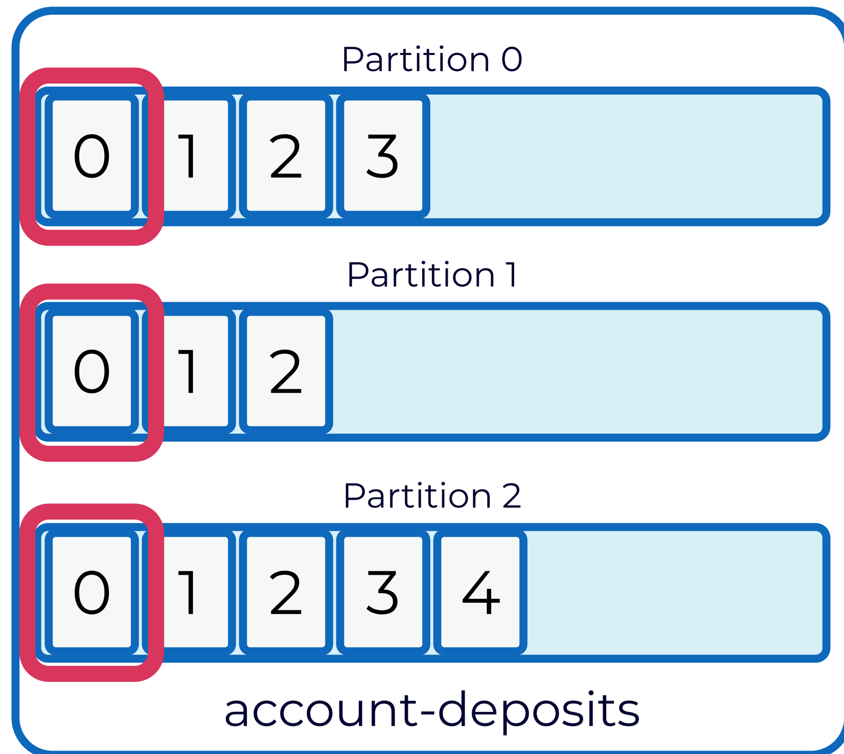
When events are written to a partition, they are assigned an offset identifying the logical position within the partition



Partition Offsets

When events are written to a partition, they are assigned an offset identifying the logical position within the partition

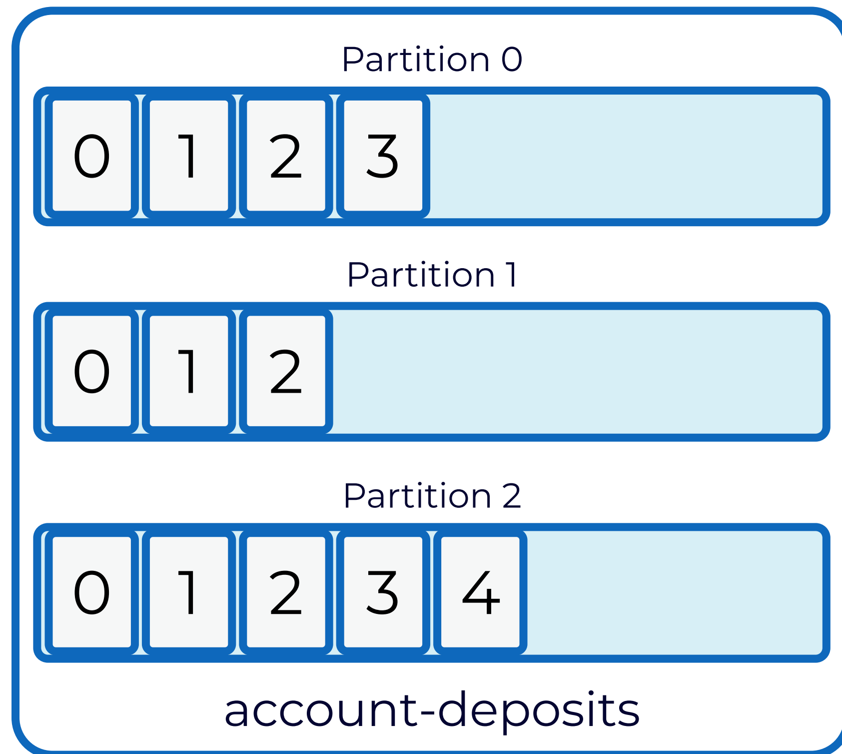
- The initial event written to each partition is assigned offset 0



Partition Offsets

When events are written to a partition, they are assigned an offset identifying the logical position within the partition

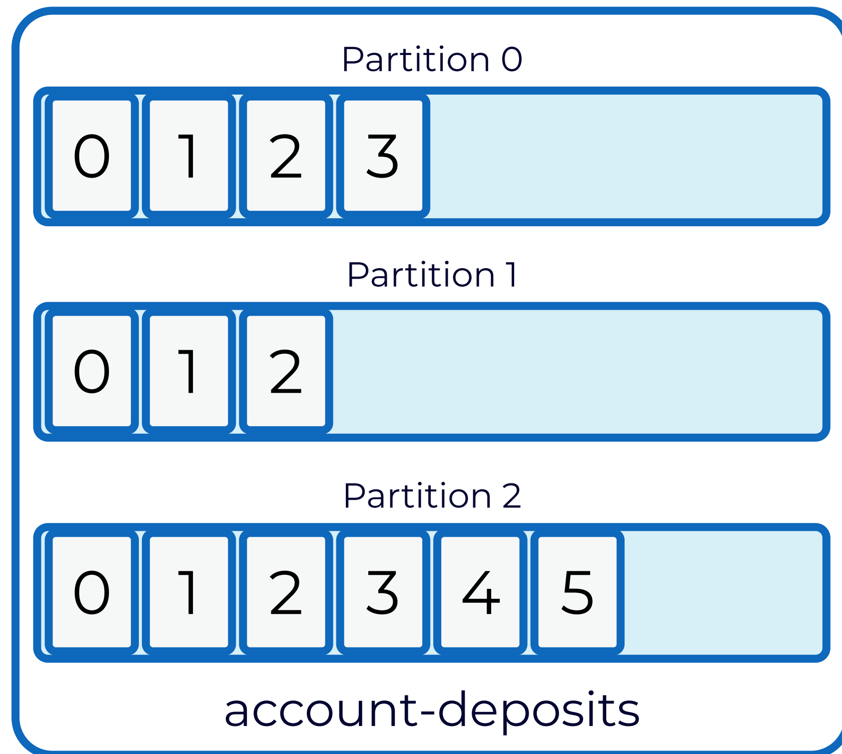
- The initial event written to each partition is assigned offset 0
- Subsequent events written to partitions are assigned the next corresponding offset for that partition



Partition Offsets

When events are written to a partition, they are assigned an offset identifying the logical position within the partition

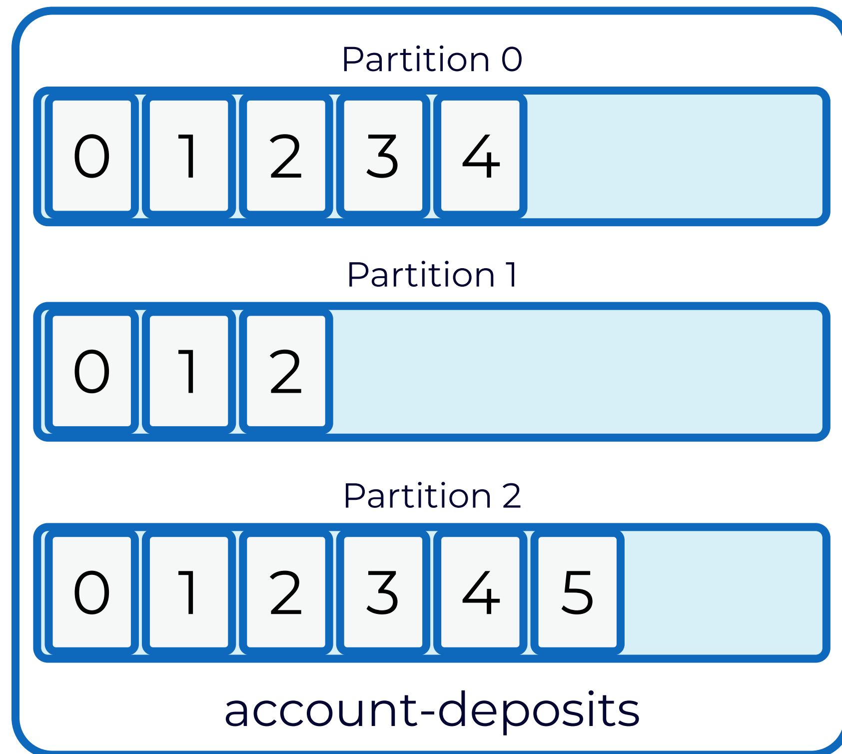
- The initial event written to each partition is assigned offset 0
- Subsequent events written to partitions are assigned the next corresponding offset for that partition



Partition Offsets

When events are written to a partition, they are assigned an offset identifying the logical position within the partition

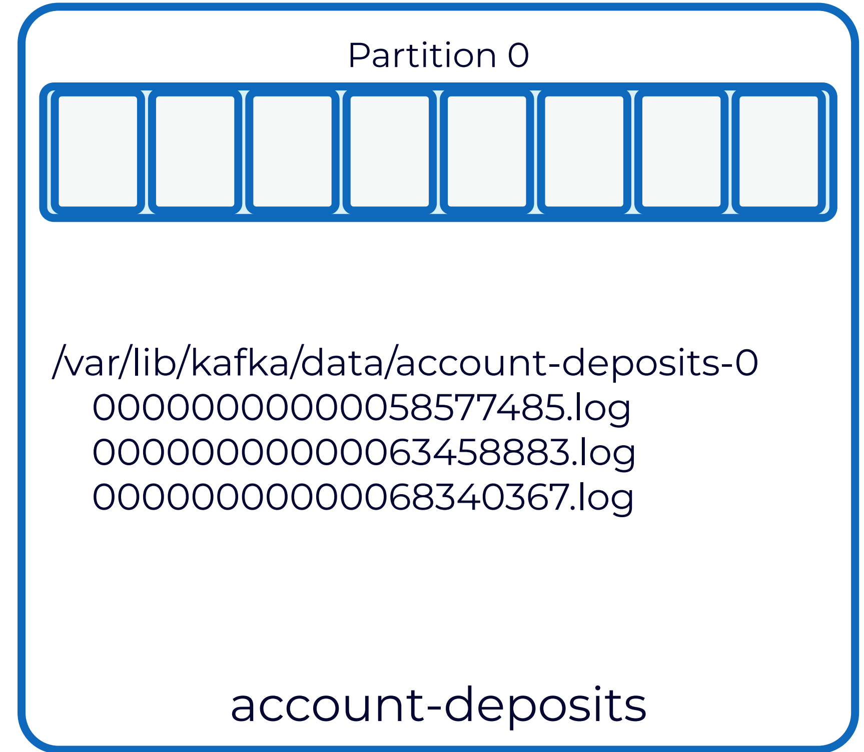
- The initial event written to each partition is assigned offset 0
- Subsequent events written to partitions are assigned the next corresponding offset for that partition



Kafka Physical Storage

Partitions exist as physical files on Kafka brokers (or in Tiered Storage)

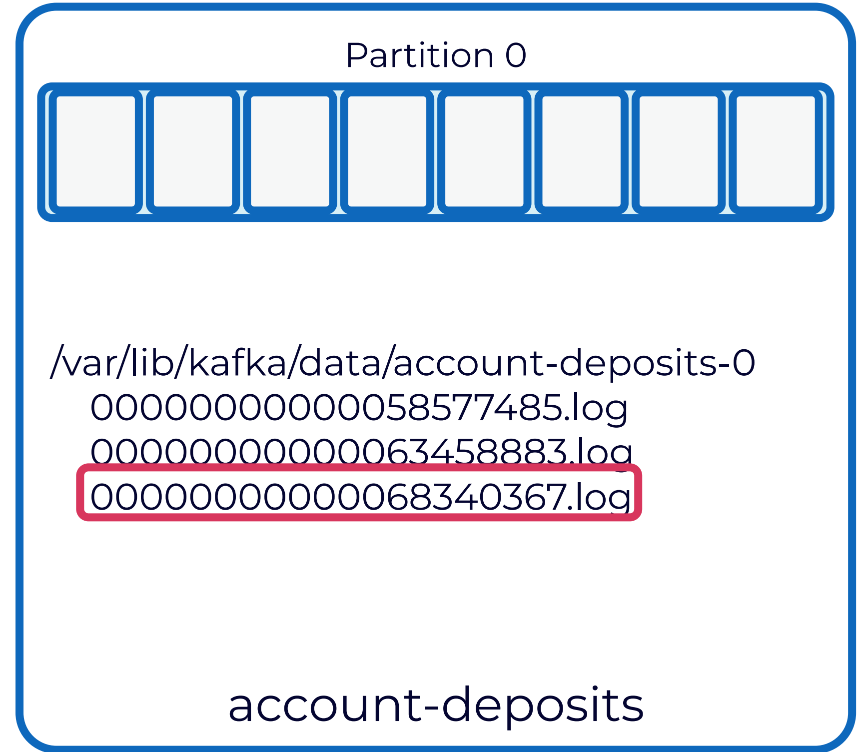
- Each partition consists of one or more log segments



Kafka Physical Storage

Partitions exist as physical files on Kafka brokers (or in Tiered Storage)

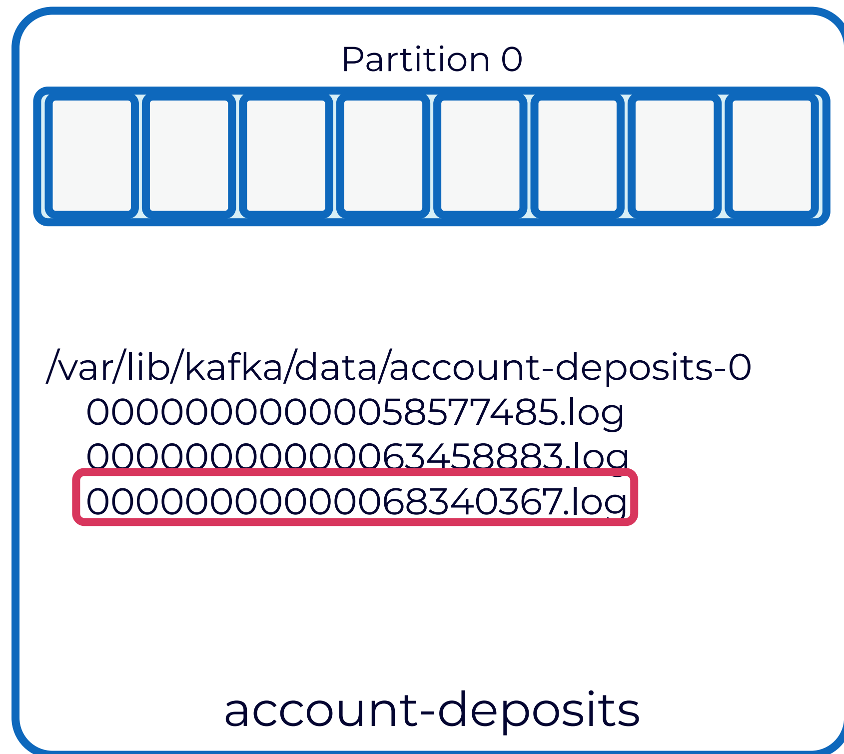
- Each partition consists of one or more log segments
- The segment that was most recently created is the active segment



Kafka Physical Storage

Partitions exist as physical files on Kafka brokers (or in Tiered Storage)

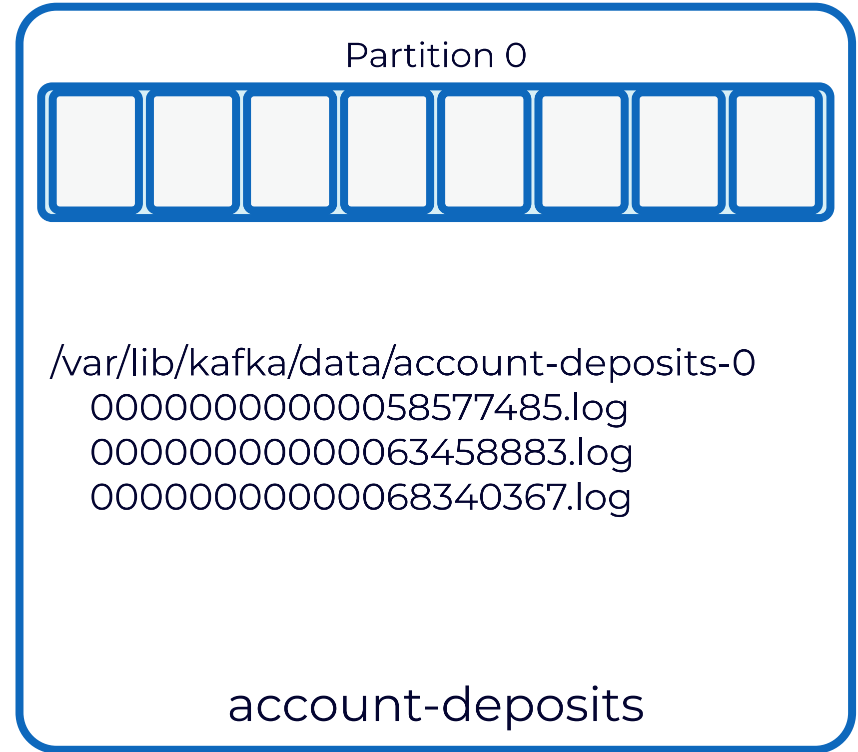
- Each partition consists of one or more log segments
- The segment that was most recently created is the active segment
  - New events are appended to the *end* of the active segment



Kafka Physical Storage

Partitions exist as physical files on Kafka brokers (or in Tiered Storage)

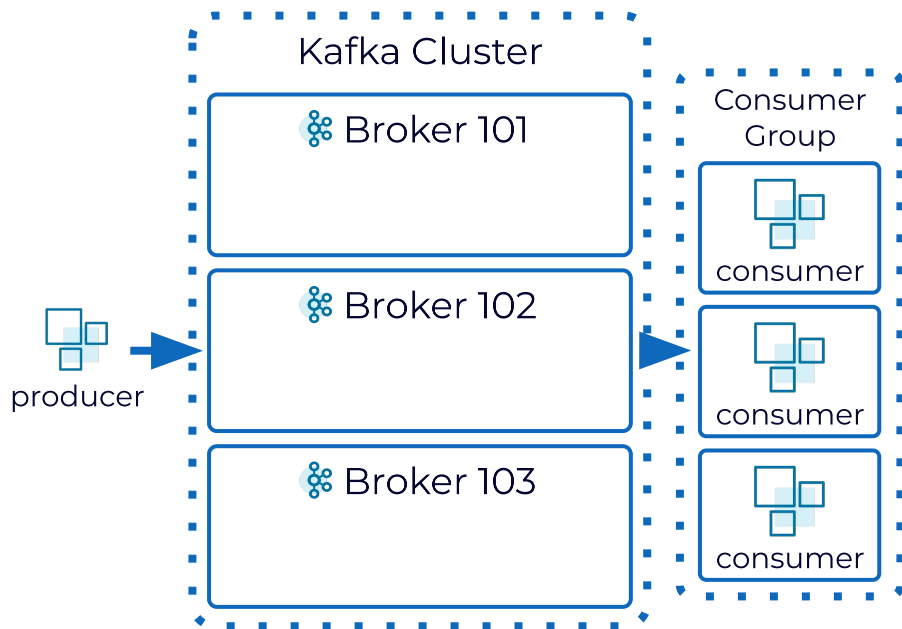
- Each partition consists of one or more log segments
- The segment that was most recently created is the active segment
  - New events are appended to the end of the active segment
- Partitions are optionally replicated to additional Kafka brokers as defined in a topic's configuration



Kafka Brokers

Kafka is composed of a network of machines called brokers

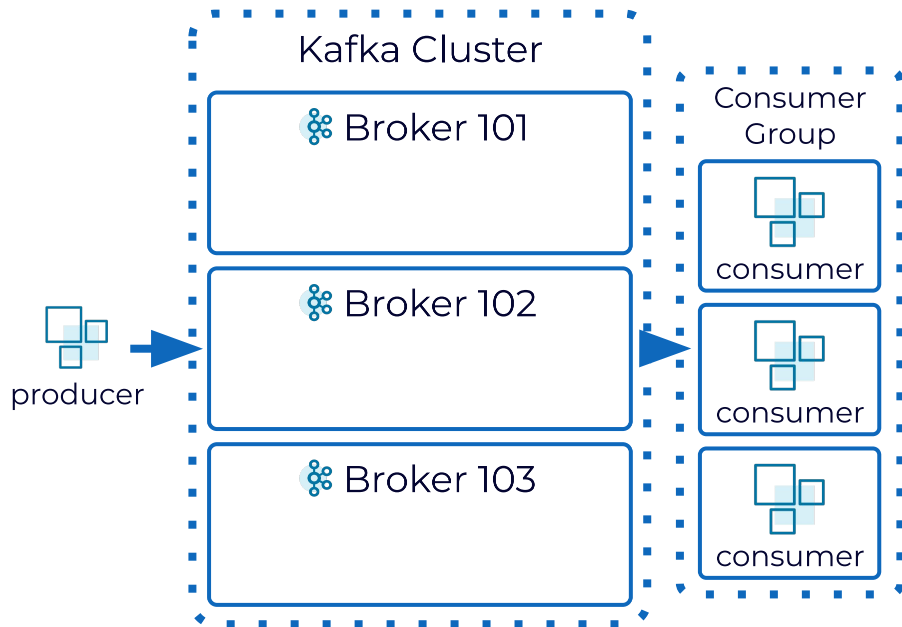
- A cloud instance, computer, or container running the Kafka process
- Form a Kafka cluster



Kafka Brokers

Kafka is composed of a network of machines called brokers

- A cloud instance, computer, or container running the Kafka process
- Form a Kafka cluster
- Manage storage of topics, partitions, and events
- Handle write and read requests



Kafka Brokers

Kafka is composed of a network of machines called brokers

- A cloud instance, computer, or container running the Kafka process
- Form a Kafka cluster
- Manage storage of topics, partitions, and events
- Handle write and read requests
- Manage replication of partitions

Kafka Cluster

Broker 101

Broker 102

Broker 103

producer

Consumer Group

consumer

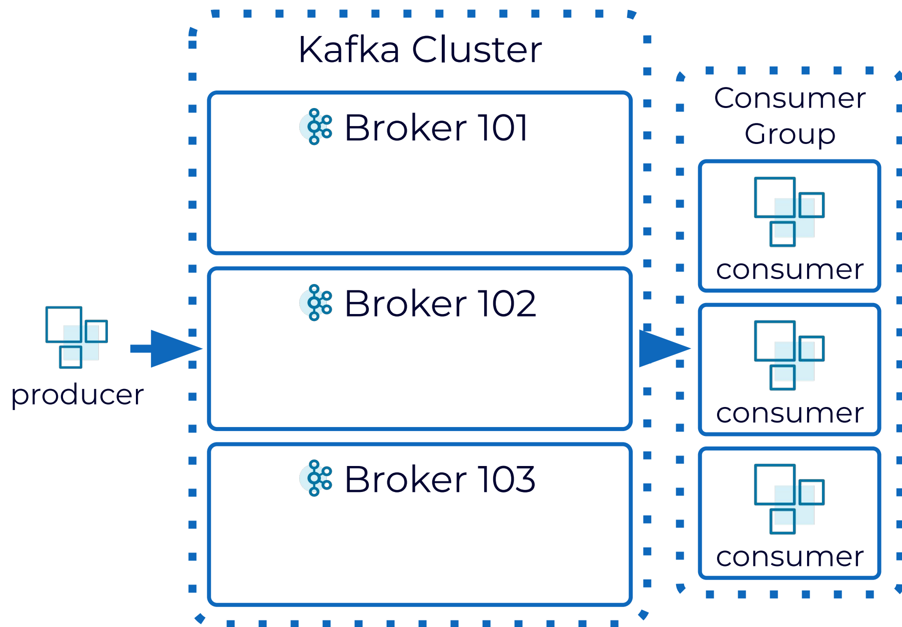
consumer

consumer

Kafka Brokers

Kafka is composed of a network of machines called brokers

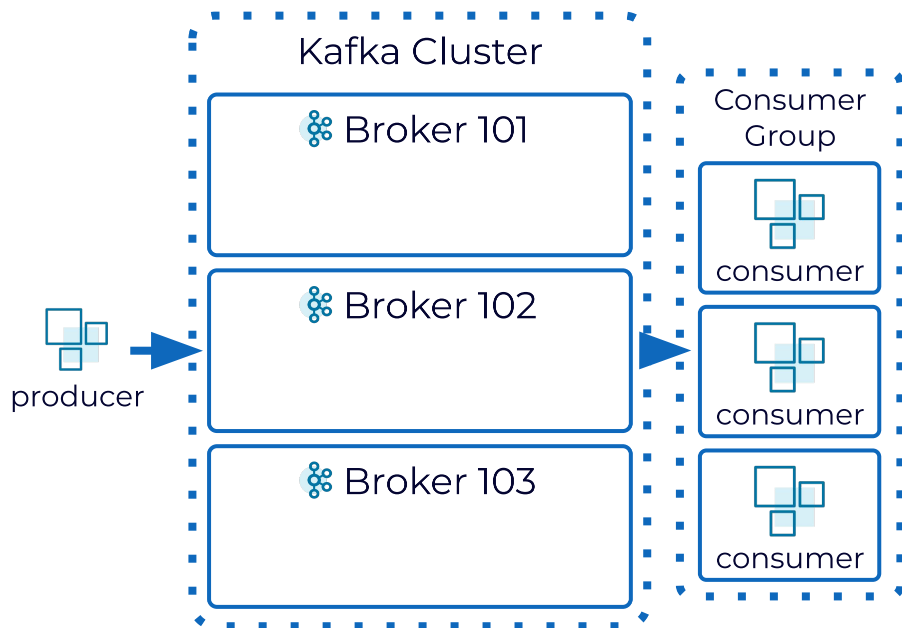
- A cloud instance, computer, or container running the Kafka process
- Form a Kafka cluster
- Manage storage of topics, partitions, and events
- Handle write and read requests
- Manage replication of partitions
- One broker, which is dynamically chosen for fault tolerance, acts as the cluster controller



Kafka Brokers

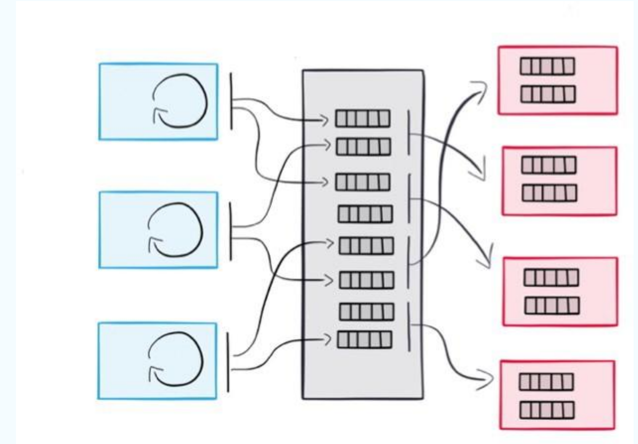
Kafka is composed of a network of machines called brokers

- A cloud instance, computer, or container running the Kafka process
- Form a Kafka cluster
- Manage storage of topics, partitions, and events
- Handle write and read requests
- Manage replication of partitions
- One broker, which is dynamically chosen for fault tolerance, acts as the cluster controller
  - We will cover this in detail in the control plane module



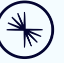
Decoupling Producers and Consumers

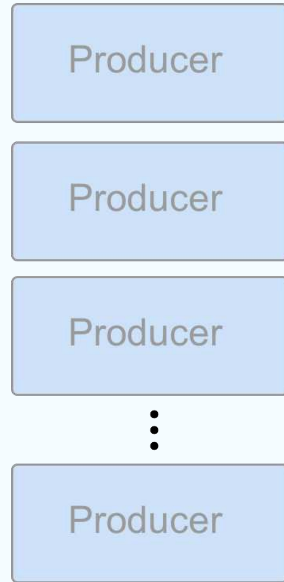
- Producers and Consumers are decoupled
- Slow Consumers do not affect Producers
- Add Consumers without affecting Producers
- Failure of Consumer does not affect System

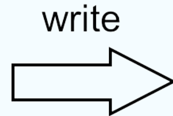


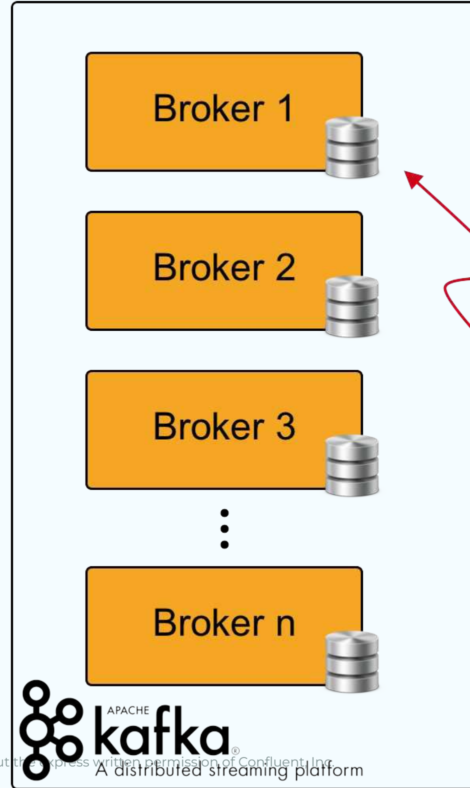
Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

Kafka Producers







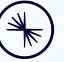


local storage

Broker:
  log.retention.hours
Topic:
  retention.ms
Default is 7 days

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

Kafka Producers



Topic1 partition1

Topic1 partition1

Topic1 partition1

P

Broker 1

Broker 2

Broker 3

Producer Properties

`acks=0`

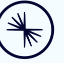
Leader

Follower

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written pe

Reference
https://www.confluent.io/blog/exactly-once-semantics-are-possible-heres-how-apache-kafka-does-it/

Kafka Producers



P

Topic1
partition1

Topic1
partition1

Topic1
partition1

ack

Broker 1

Broker 2

Broker 3

Producer Properties

`acks=1`

Leader

Follower

Reference
https://www.confluent.io/blog/exactly-once-semantics-are-possible-heres-how-apache-kafka-does-it/

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written pe

Kafka Producers

Topic1
partition1

Topic1
partition1

Topic1
partition1

P

ack

Broker 1

Broker 2

Broker 3

Producer Properties

```
acks=all
min.insync.replica=2
```

Leader

Follower

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written pe

Kafka Producers

```
{key: 1234 data: abcd} - offset 3345
```

Successful write

Topic1 partition1

Topic1 partition1

Topic1 partition1

P

Failed ack

Broker 1

Broker 2

Broker 3

Producer Properties

```
acks=all
```

Leader

Follower

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written pe

Kafka Producers

```
{key: 1234, data: abcd} - offset 3345
{key: 1234, data: abcd} - offset 3346
```

dupe!

retry

P

ack

Topic1
partition1

Topic1
partition1

Topic1
partition1

Broker 1

Broker 2

Broker 3

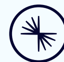
Producer Properties

`acks=all`

Reference
https://www.confluent.io/blog/exactly-once-semantics-are-possible-heres-how-apache-kafka-does-it/

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written pe

Kafka Producers



```
(pid, seq) [payload]
(100, 1) {key: 1234, data: abcd} - offset 3345
(100, 1) {key: 1234, data: abcd} - rejected, ack re-sent
(100, 2) {key: 5678, data: efgh} - offset 3346
```

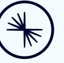

Producer Properties

```
acks=all
enable.idempotence=true
max.inflight.requests.per.connection=5
retries > 0
```

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written pe

Reference
https://www.confluent.io/blog/exactly-once-semantics-are-possible-heres-how-apache-kafka-does-it/

Kafka Consumers



Broker 1

Broker 2

Broker 3

⋮

Broker n

poll

repeating forever!

Consumer

Data Records

Reports

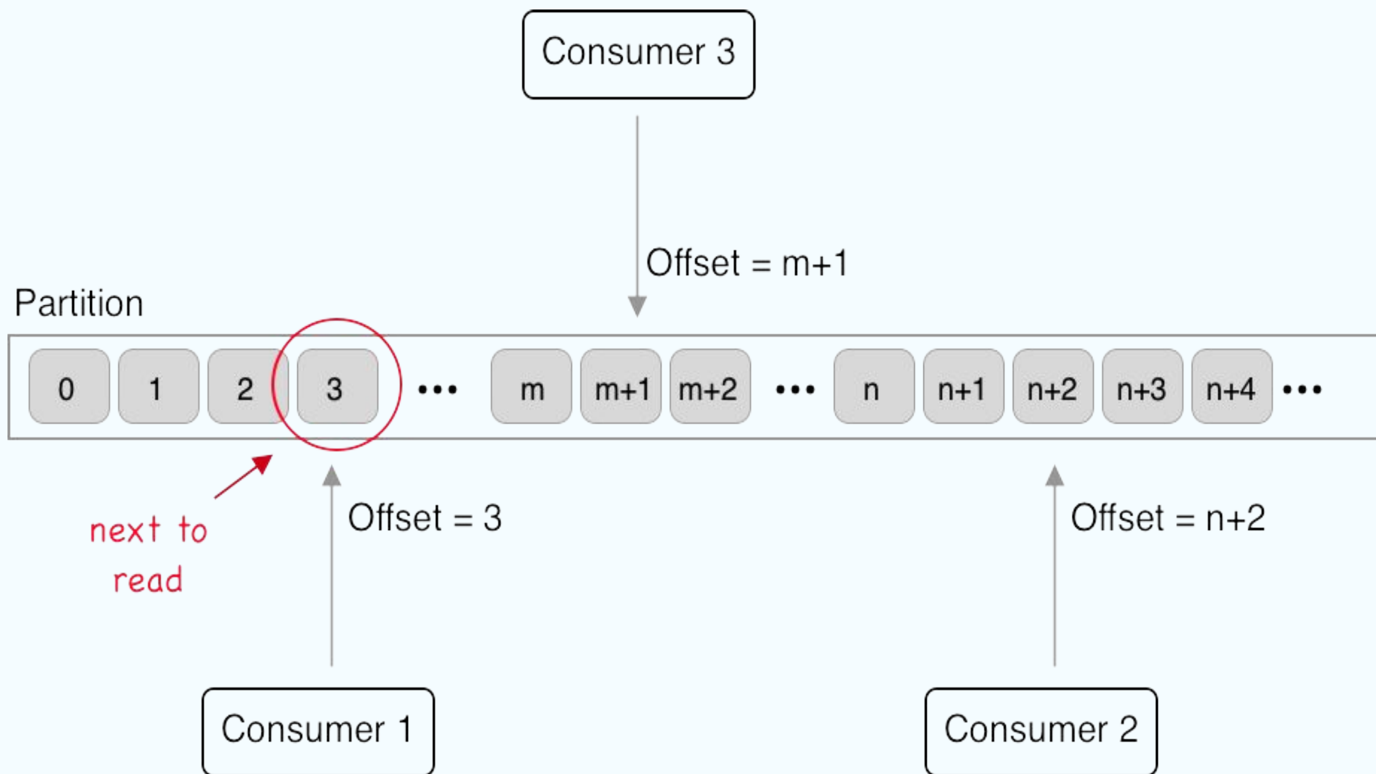
Dashboards

Insight

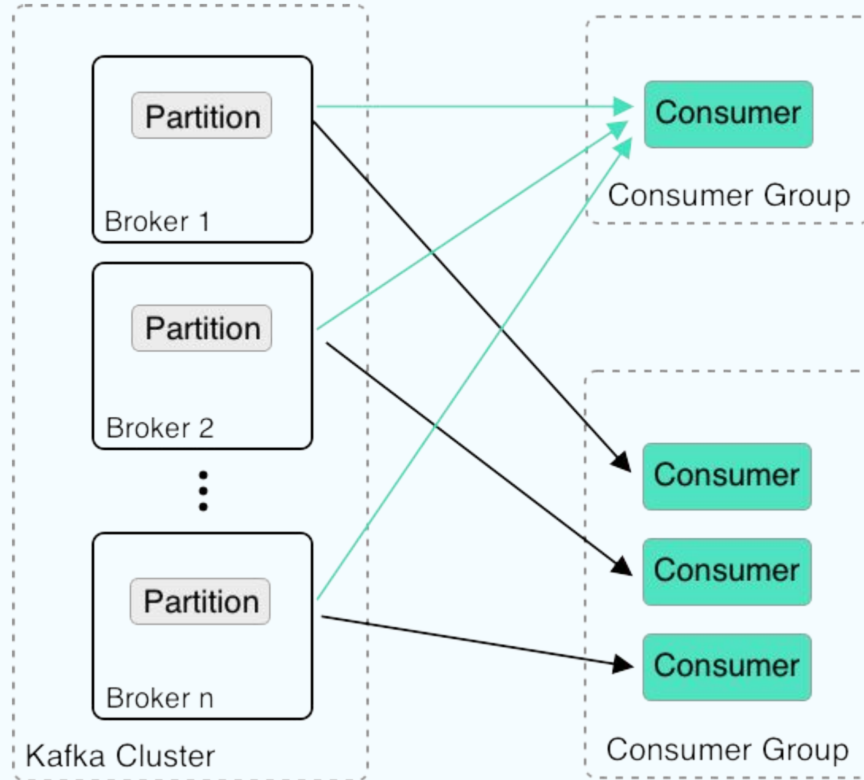
Processes

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

Consumers have a position of their own



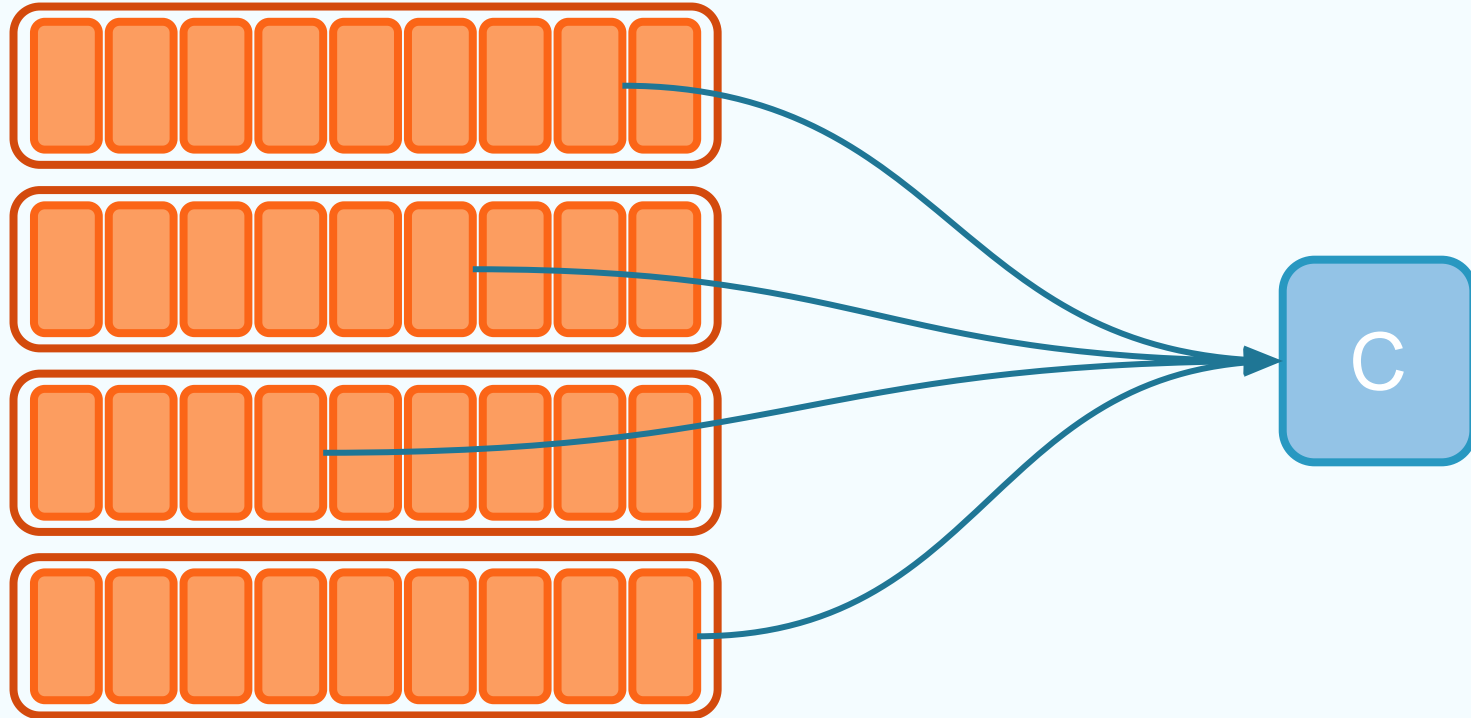
Consumers have a position of their own



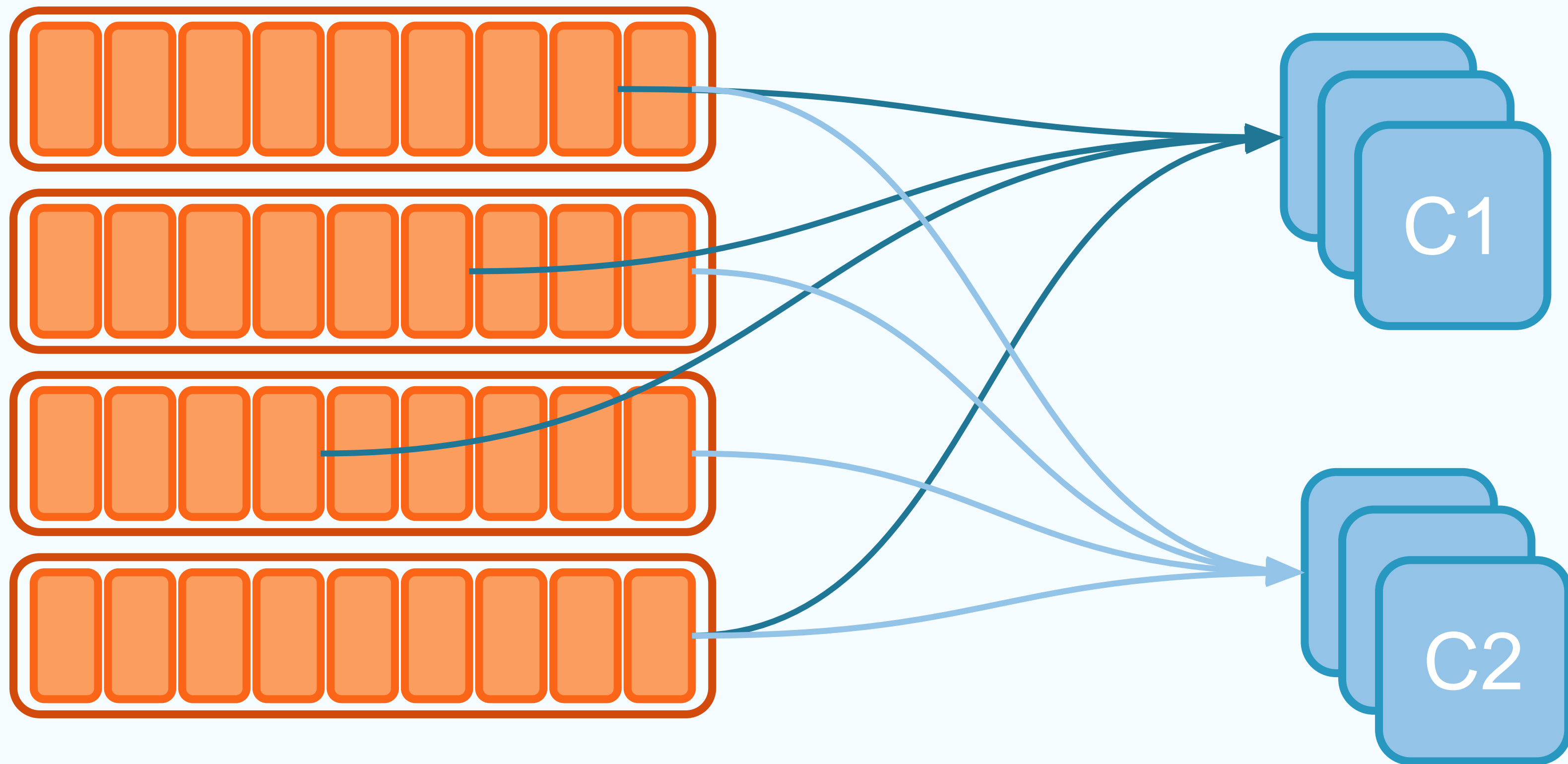
A basic Java consumer

```
final Consumer<String, String> consumer = new KafkaConsumer<String, String>(props);
consumer.subscribe(Arrays.asList(topic));
try {
  while (true) {
    ConsumerRecords<String, String> records = consumer.poll(100);
    for (ConsumerRecord<String, String> record : records) {
      -- Do Some Work --
    }
  }
} finally {
  consumer.close();
}
}
```

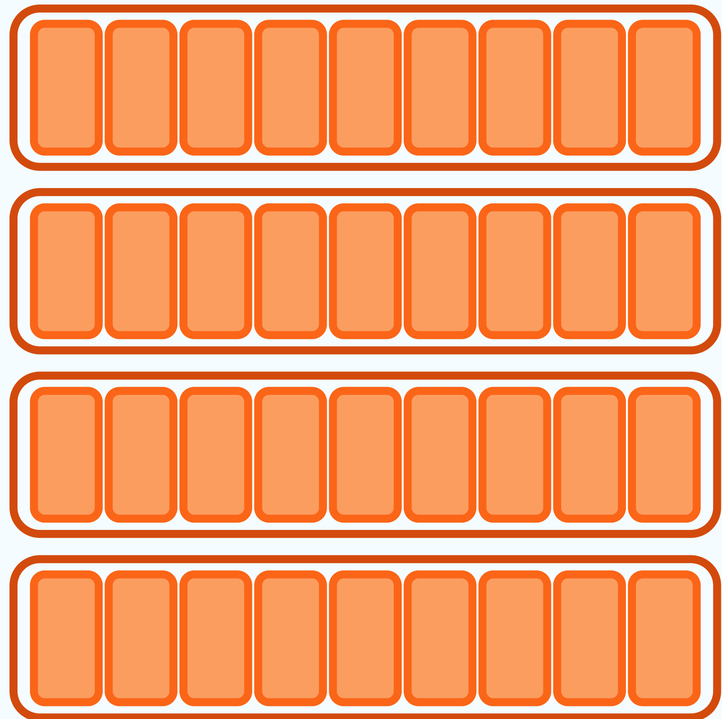
A basic consumer

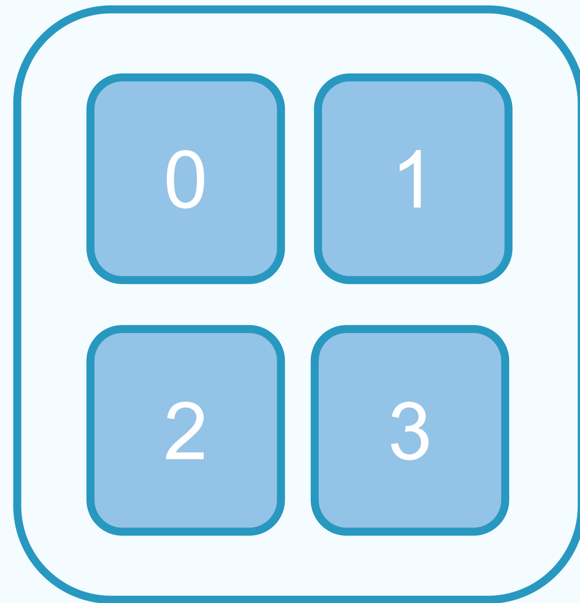


Group consumption

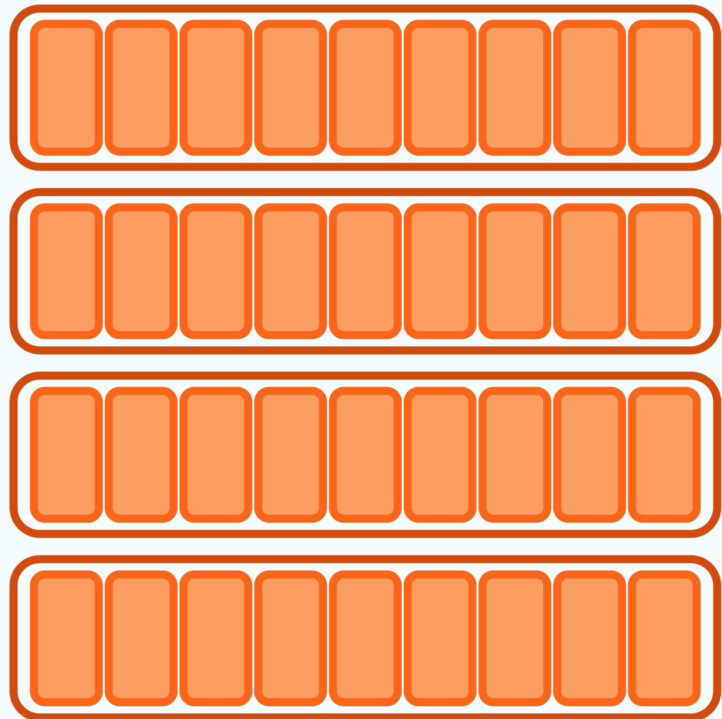


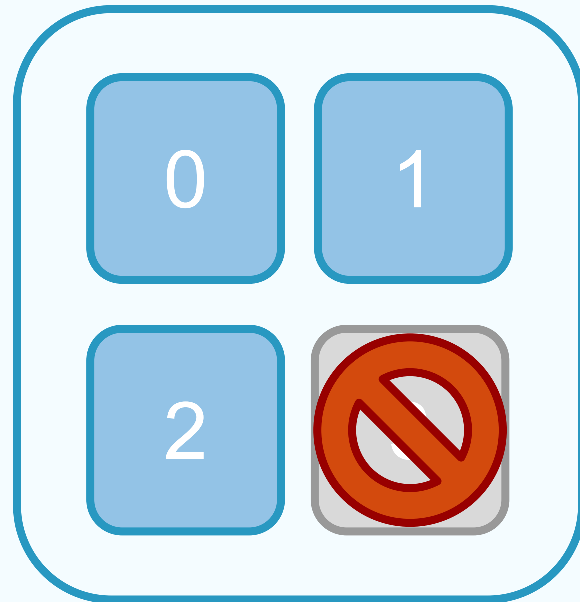
Group consumption



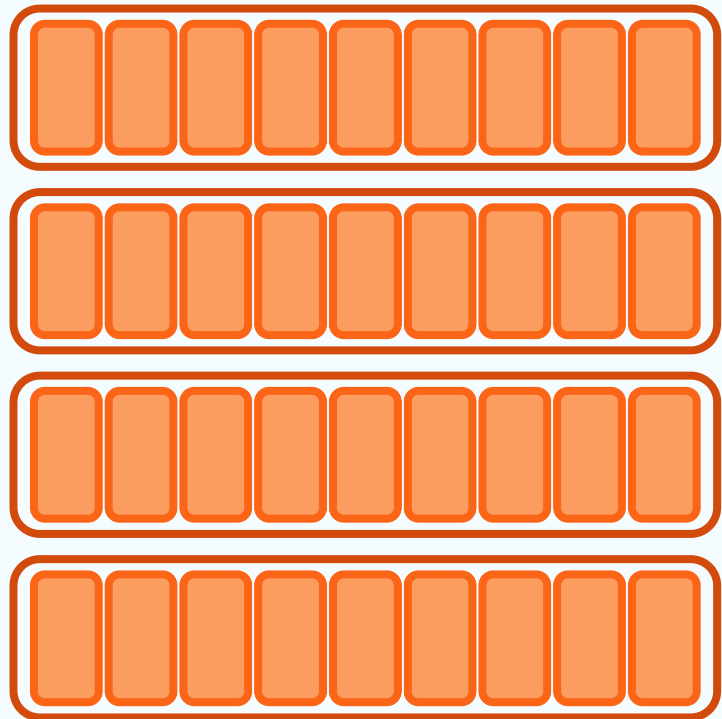


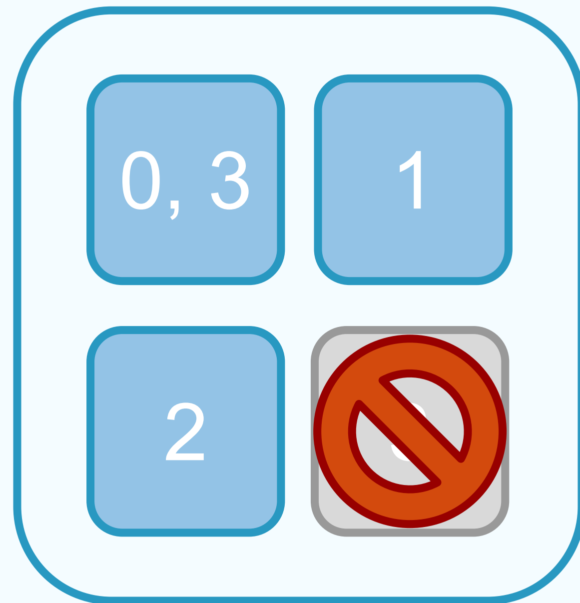
Group consumption





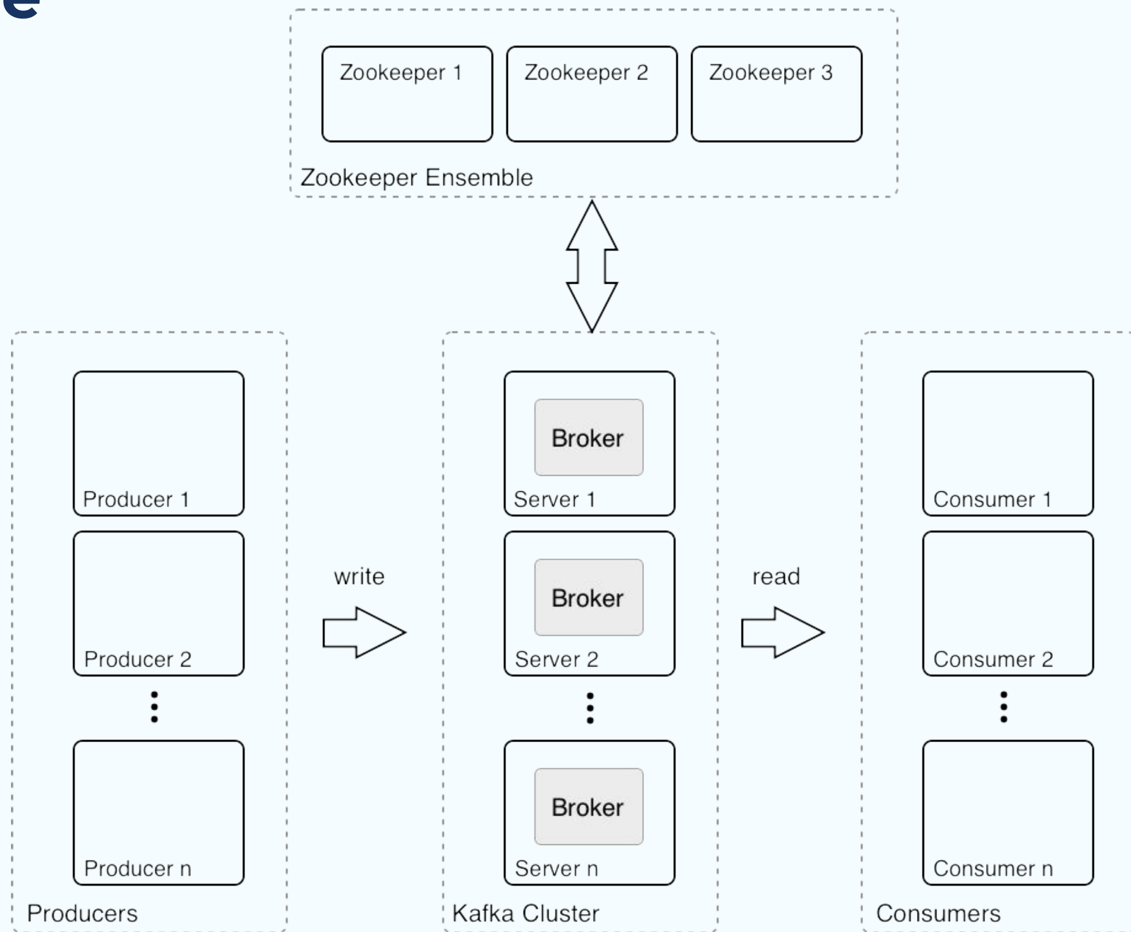
Group consumption



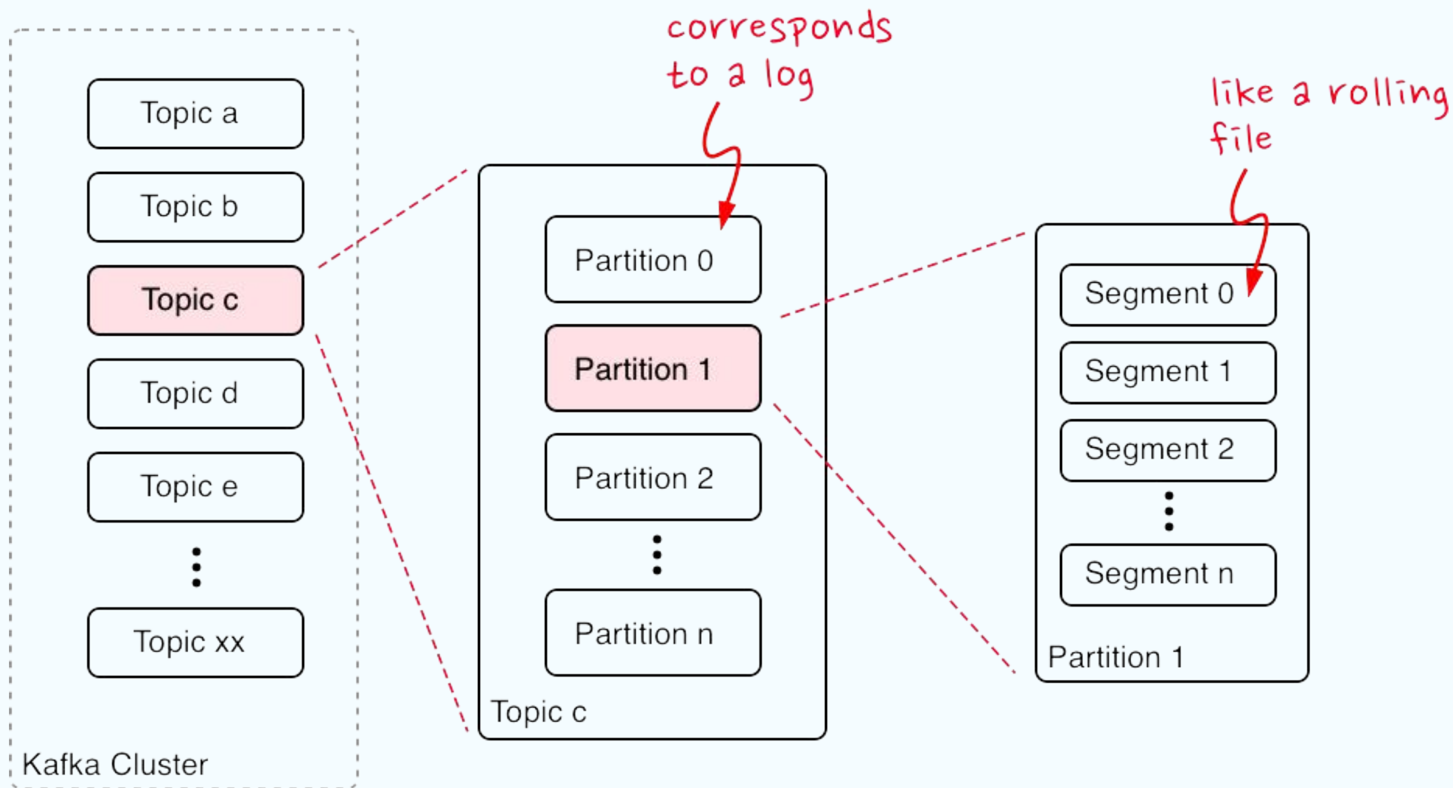


Kafka Architecture

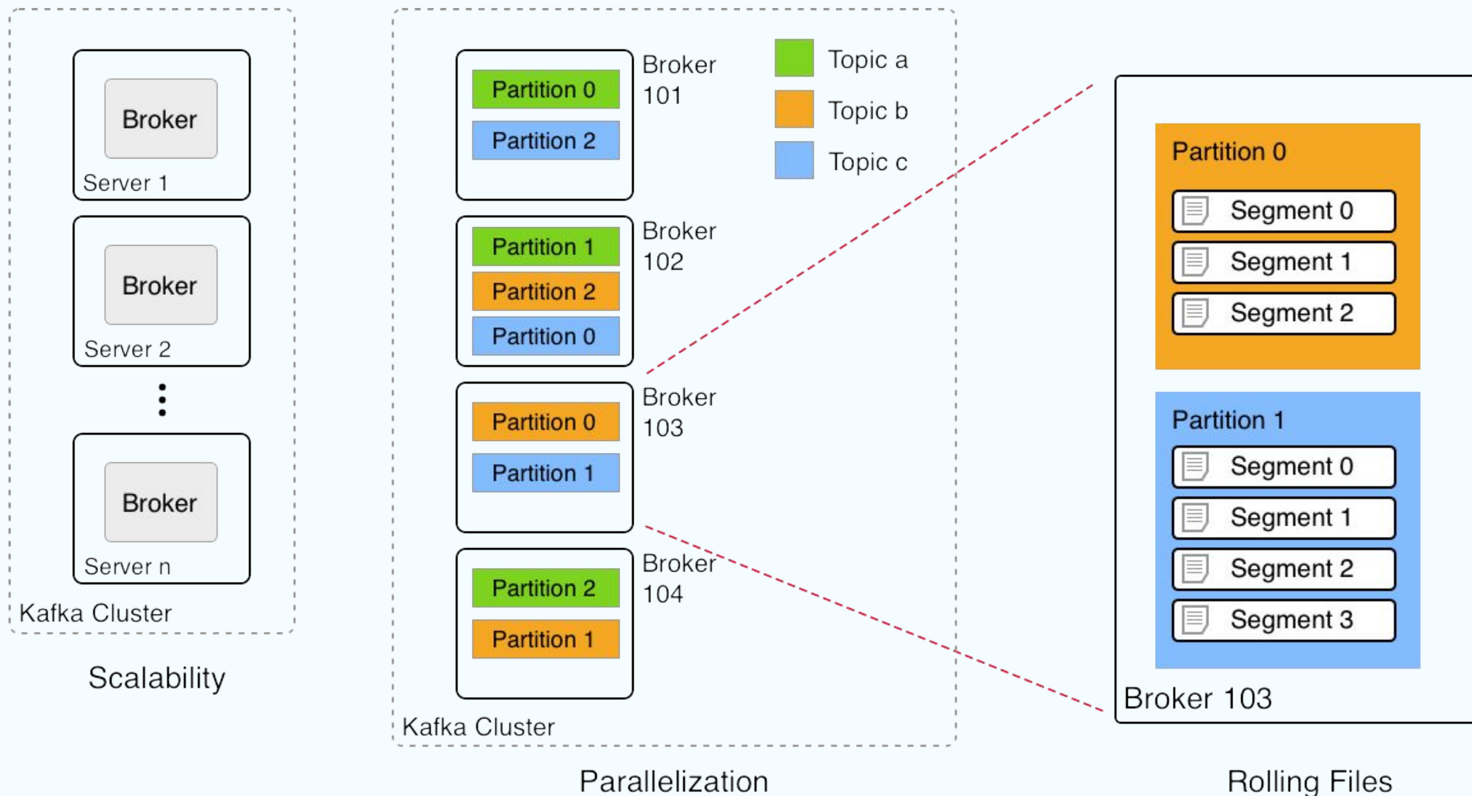
Architecture



Topic, Partitions and Segments



Topic, Partitions and Segments



Physical layout of kafka logs

```
$ tree freblogg*
freblogg-0
|-- 00000000000000000000.index
|-- 00000000000000000000.log
|-- 00000000000000000000.timeindex
`-- leader-epoch-checkpoint
freblogg-1
|-- 00000000000000000000.index
|-- 00000000000000000000.log
|-- 00000000000000000000.timeindex
`-- leader-epoch-checkpoint
freblogg-2
|-- 00000000000000000000.index
|-- 00000000000000000000.log
|-- 00000000000000000000.timeindex
`-- leader-epoch-checkpoint
```

Processing

Filter Events to a Separate Stream in Real Time

Stream: Blue and Red Events

Partition 0

Partition 1

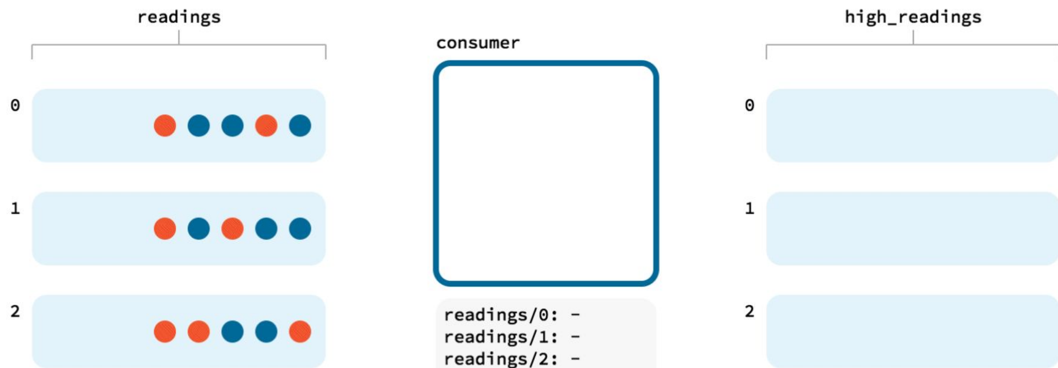
Partition 2

Stream: Blue Events Only

Partition 0

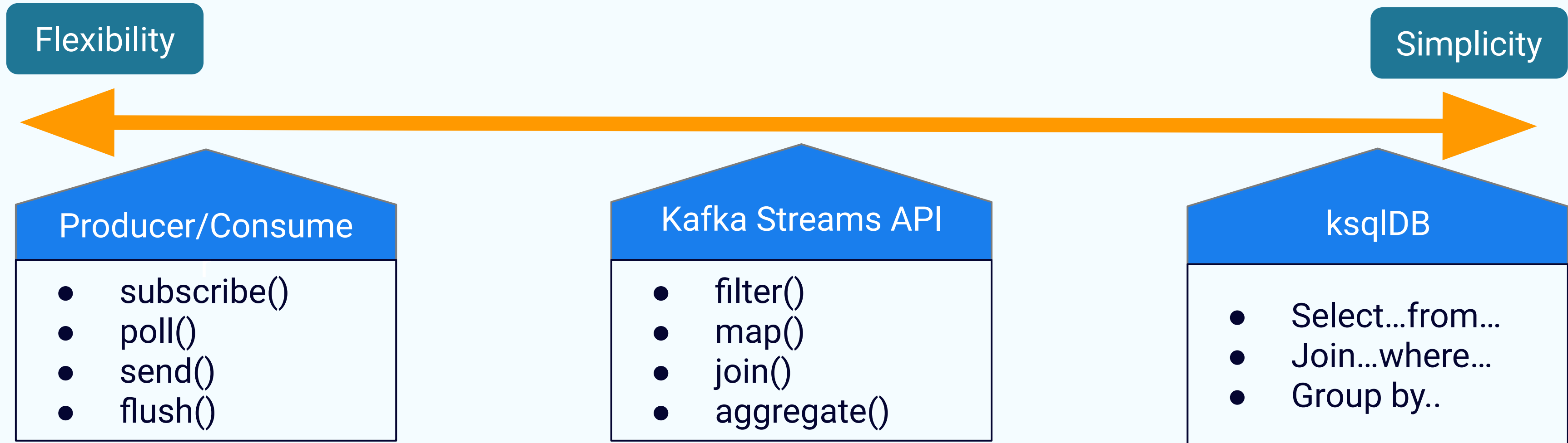
Partition 1

Partition 2

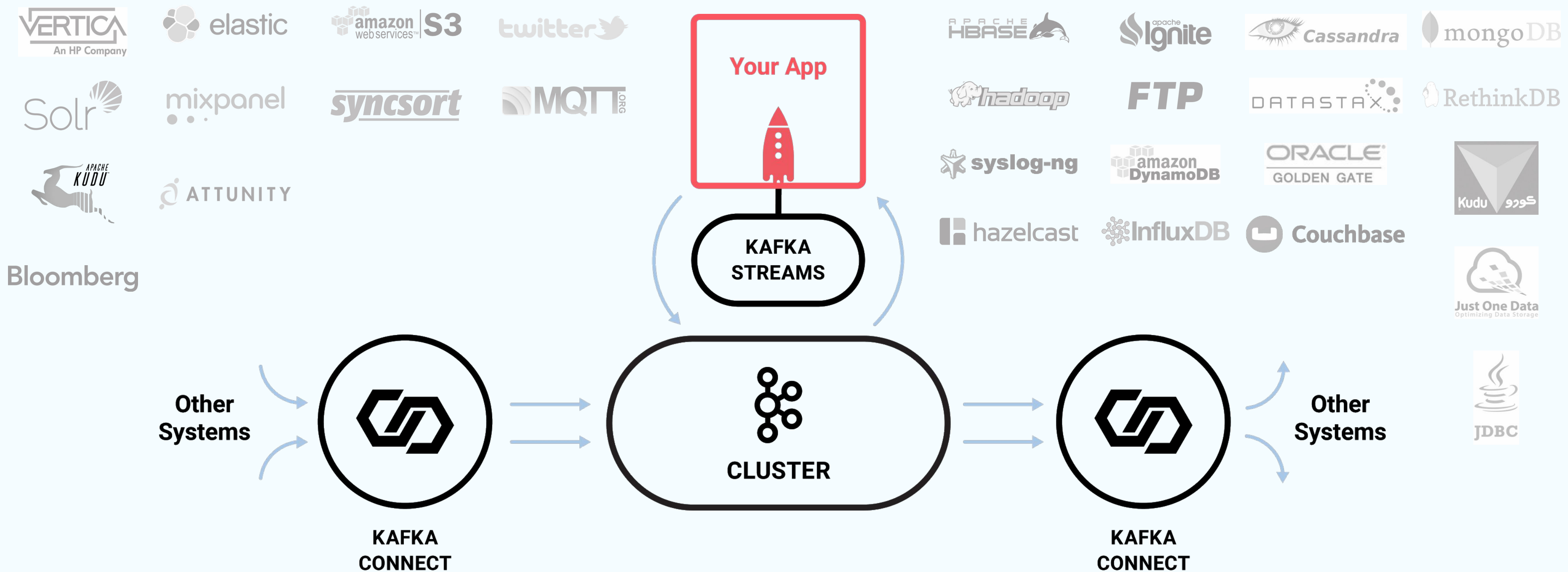



```sql
-- pq1
CREATE STREAM high_readings AS
  SELECT sensor,
         reading,
         UCASE(location) AS location
  FROM readings
  WHERE reading > 41
  EMIT CHANGES;
```

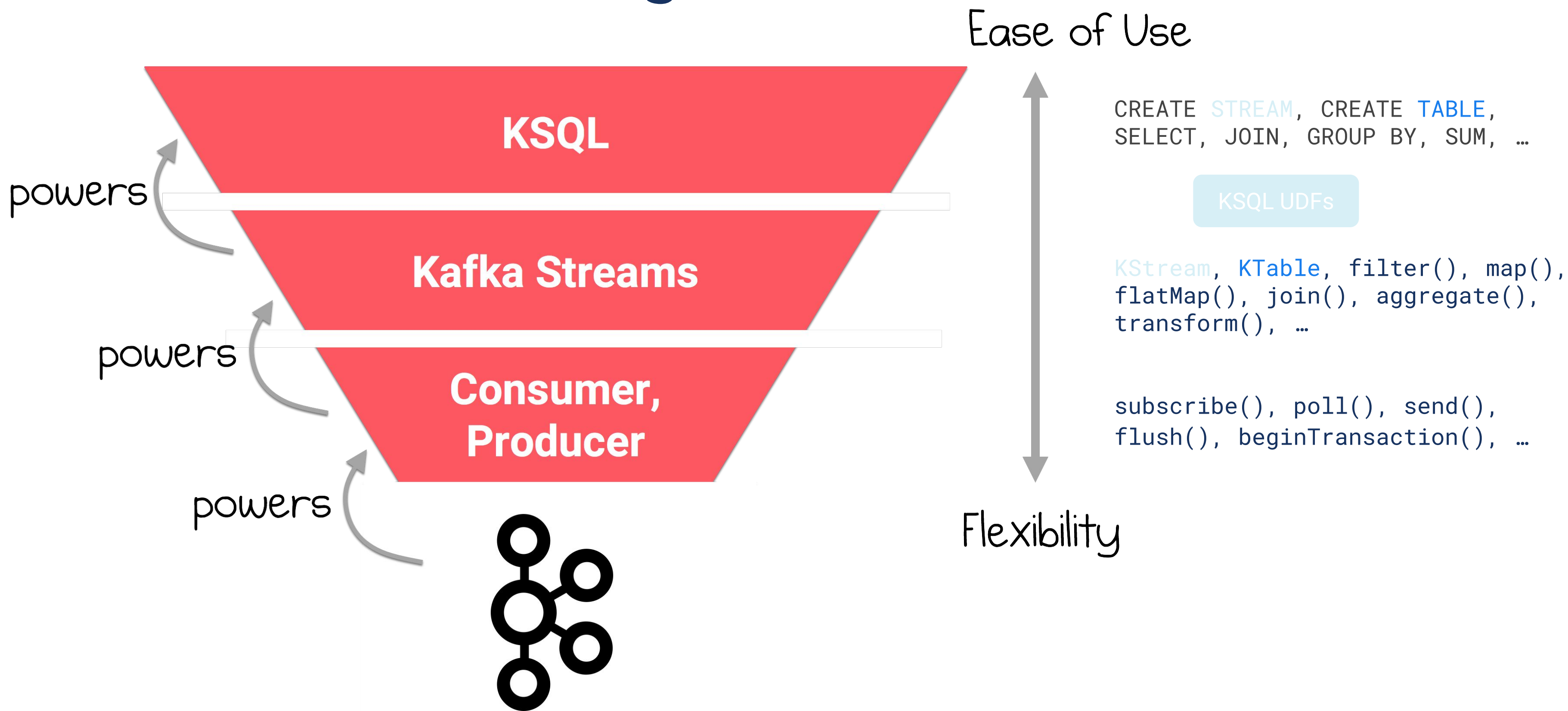
Connect All Applications and Data Sources and Sinks



Connect All Applications and Data Sources and Sinks



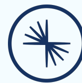
Shoulders of Streaming Giants



Stream Processing

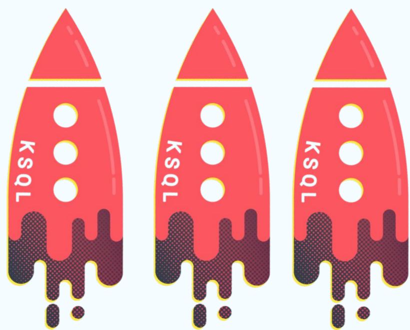
is the ***toolset*** *for dealing with* ***events***

as ***they move!***

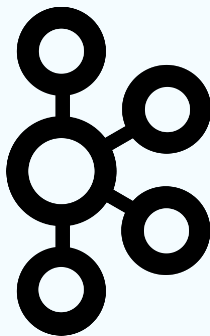


Interaction with Kafka

ksqlDB
(processing)



Kafka
(data)



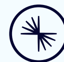
JVM application
with Kafka Streams (processing)

Does not run on
Kafka brokers

Does not run on
Kafka brokers

Generating random data for Kafka

Generating random traffic for Kafka



- **Different** solutions
    - Datagen (Kafka connect based) is the official solution
        - Needs a Kafka connect environment (not immediate to setup)
        - In the managed version, can’t be customised with your data
        - In the managed version, can’t for example do **compression**
        - Not managing real **relations** between data
    - There are other tools
        - Not managing relations, **or** complex to use, **or** abandoned **or** not flexible enough

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> apropos jr

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> apropos jr

- **J**son **R**andom generator
- **J**ust another **R**andom generator
- Similar to **JQ**, which is one of the tools I use most
  https://stedolan.github.io/jq/
- ...

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> apropos jr

- **J**son **R**andom
- **J**ust another **R**
- Similar to **JQ**, w
  https://stedolar
- ...

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> history | grep jr

- Had to generate traffic for a customer, on the fly, with just an **example** of a json
- They asked how much this stuff would be **compressed** by the producer, which obviously varies with:
  - different **algorithms**
  - different **throughput**
  - different **batching** kafka configuration
  - can’t use a single json to do that, would be compressed **too much**
- Existing tools couldn’t easily answer this question, and for sure not in a 5 minutes time frame, for example:
  - **Datagen** with custom objects is complex to setup
  - Managed **Datagen** on Confluent Cloud can’t use custom objects and can’t compress

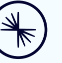
Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> history | grep jr

```
{
"VLAN": "DELTA",
"IPV4_SRC_ADDR": "10.1.41.98",
"IPV4_DST_ADDR": "10.1.137.141",
"IN_BYTES": 1220,
"FIRST_SWITCHED": 1681984281,
"LAST_SWITCHED": 1682975009,
"L4_SRC_PORT": 81,
"L4_DST_PORT": 80,
"TCP_FLAGS": 0,
"PROTOCOL": 1,
"SRC_TOS": 211,
"SRC_AS": 4,
"DST_AS": 1,
"L7_PROTO": 443,
"L7_PROTO_NAME": "ICMP",
"L7_PROTO_CATEGORY": "Application"
}
```


Copyright 2021, Confluent, Inc. All rights reser

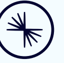
> history | grep jr



```
{
"VLAN": "{{randoms "ALPHA|BETA|GAMMA|DELTA"}}",
"IPV4_SRC_ADDR": "{{ip "10.1.0.0/16"}}",
"IPV4_DST_ADDR": "{{ip "10.1.0.0/16"}}",
"IN_BYTES": {{integer 1000 2000}},
"FIRST_SWITCHED": {{unix_time_stamp 60}},
"LAST_SWITCHED": {{unix_time_stamp 10}},
"L4_SRC_PORT": {{ip_known_port}},
"L4_DST_PORT": {{ip_known_port}},
"TCP_FLAGS": 0,
"PROTOCOL": {{integer 0 5}},
"SRC_TOS": {{integer 128 255}},
"SRC_AS": {{integer 0 5}},
"DST_AS": {{integer 0 2}},
"L7_PROTO": {{ip_known_port}},
"L7_PROTO_NAME": "{{ip_known_protocol}}",
"L7_PROTO_CATEGORY": "{{randoms "Network|Application|Transport|Session"}}"
}
```

Copyright 2021, Co

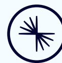
> whois jr



- Is a **template** system, leveraging wonderful Golang **text/template** package
- Has a **CLI** but also **REST APIs** (in beta)
- Can generate **anything** you could write a template for (so, not really tied to json)
- Embeds a specialized **fake** library (no use of existing faking libraries)
- Has **automatic integrity** for related fields (city, zip, mobile, phone, email/company, etc)
- Can maintain **integrity** between objects generated (**relations**)
- It's been designed for **Kafka**, but can directly output to **Elastic**, **Redis**, **MongoDB, S3**
- Can talk to **Confluent Schema Registry** for Kafka, serializing in **Avro/Json Schema**

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> man jr



- You choose your **template** from the available templates
- You choose **-n** number of objects to generate at each pass
- You choose **-f** frequency
- You choose **-d** duration

jr template **list**
jr template **run** *net_device* | jq
jr template **run** -n 2 *net_device* | jq
jr template **run** -n 2 -f 100ms *net_device* | jq
jr template **run** -n 2 -f 100ms -d 5s *net_device* | jq

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> man template

- There are **3** different templates to control jr
    - **Key** template, which defaults to **null**
    - **Output** template, which defaults to **Value** only: **{{.V/n}}**
    - **Value** template, which you control in two different ways
        - Embedding directly in the command line (**--embedded**)
        - By name (**user**,**net_device**, etc) for the OOTB templates

jr template **list**
jr template **show** *net_device*
jr template **show** *user*
jr template **run** --key '{{key "ID" 100}}' *user*
jr template **run** --key '{{key "ID" 100}}' --outputTemplate '{{.K}} {{.V}}' *net_device*
jr template **run** --key '{{key "ID" 100}}' --embedded '{{name}} {{email}}' --kcat

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> cat cli

- You have 3 resources: **emitters**, **templates** and **functions**
  - You can list, show and run **templates**
  - You can list available **functions** and test directly (**--run**) without writing a template. There are **126** functions at the moment, and growing
  - **Emitters** are a new concept: you configure different emitters all at once, with different frequency and other parameters, and then you just list/show/run the emitters with a single command

jr function **list** -c finance
jr function **list** card --run
jr function **list** regex --run
jr emitter **list**
jr emitter **run**

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

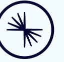
> man functions

- There are **126** functions at the moment, categorized as
  - People
  - Text utilities
  - Network
  - Context
  - Address
  - Finance
  - Math
  - Phone

```
cat .jr/templates/data/it/movie
jr template run --template '{{from "movie"}}'
jr template run --locale IT --template '{{from_n "beer" 3}}'
jr template run --locale IT --template '{{from_n "actor" 15}}'
```

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> cat automatic_integrity

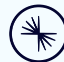


- Some functions are “smart”, for example:
  - **Mobile** phones are generated by “inverse” regular expressions, using mobile company numbers valid for the chosen country (**--locale**)
  - Streets, cities, zip codes, phone prefix and more are all **localizable** and **coherent** without doing anything special
  - your **work email** is generated automatically using - if already in the template - previously generated **name**, **surname** and **company**

```
jr template run --template '{{name}} {{email}}'
jr template run --template '{{name}} {{surname}} {{company}} {{email_work}}'
jr template run user | jq
jr --locale IT template run user | jq
jr --locale FR template run user | jq
```

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> echo “hello” 2>&1 >> $LOG

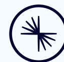


- You can choose different **output** for jr:
  - **stdout** (default)
  - **kafka**
  - **redis**
  - **mongo**
  - **elastic**
  - **s3**
- Each **output** needs a specific configuration
- Output can easily be extended implementing **Producer** interface

jr template **run** user -o kafka
jr template **run** user -o kafka -t topic_user -a
jr template **run** user -o mongo

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

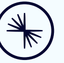
> select * from customers where custID='X1001';



- **Relational Integrity** is where most of similar tools fall. To generate “related” data, they end up having long lists of prebuilt json documents, not at all random. Basically they become equivalent to:
  - **kcat** -P -b localhost:9092 -t topic -K: -l **prebuilt_json.txt**
- jr has two features to help with integrity
  - **preload** to create a bunch of events at the beginning
  - context functions, especially **add_v_to_list, random_n_v_from_list** and **random_v_from_list**

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> select * from customers where custID='X1001';



- With preload and context you can for example:
  - generate **1000** random products all at once to a topic
  - generate **100** random customers all at once and then add **1** customer every minute
  - stream **5** random orders every **100ms** by **existing** customers with **existing** products
- To test your streaming apps (**KStream**, **ksqlDB**, **Flink**), you definitely need relations!

jr function **list** -c context
jr template **show** shoe
jr template **show** shoe_customer
jr template **show** shoe_order
jr template **show** shoe_clickstream
jr emitter **run**

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> more | grep future

- We need your help!
  - Close issues if you can: https://github.com/ugol/jr/issues
  - **Localizations** in different languages
  - Useful new **functions** for templates
  - Useful pre-configured **emitters** for complex use cases
  - New **output** Producers (every k/v store is a candidate)
- Pls **star, watch** and **fork** the project on Github!
  - ~~The **brew** guys told us that we need a minimum of:~~
  - ~~**30** forks~~
  - ~~**30** watchers~~
  - ~~**75** stars~~
  - ~~(if you want to *brew install jr*!)~~

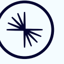
Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

> more | grep links

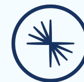
- Links
  - Issues https://github.com/ugol/jr/issues
  - Documentation https://jrnd.io/
  - Blog first part: https://dev.to/ugol/jr-quality-random-data-from-the-command-line-part-i-5e90
  - Blog second part: https://dev.to/ugol/jr-quality-random-data-from-the-command-line-part-ii-3nb3
  - Blog third part: **SOON**

Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

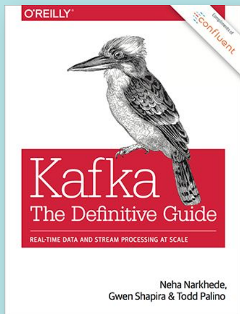
> more | grep questions?

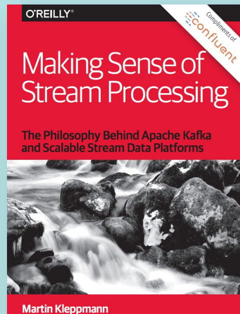


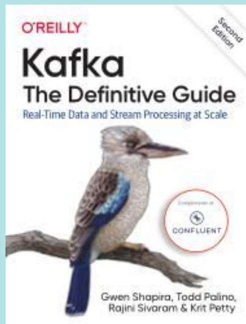
Copyright 2021, Confluent, Inc. All rights reserved. This document may not be reproduced in any manner without the express written permission of Confluent, Inc.

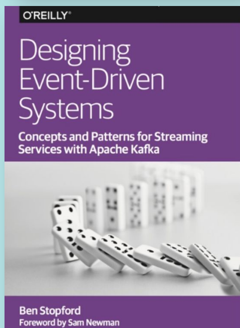


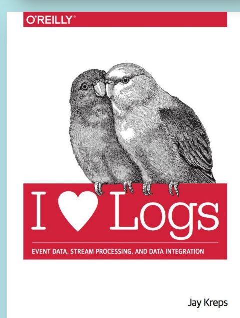
Free eBooks











Designing Event-Driven Systems
Ben Stopford

Kafka: The Definitive Guide
Neha Narkhede, Gwen Shapira, Todd Palino, I and II Edition

Making Sense of Stream Processing
Martin Kleppmann

I ❤️ Logs
Jay Kreps

http://cnfl.io/book-bundle

CONFLUENT