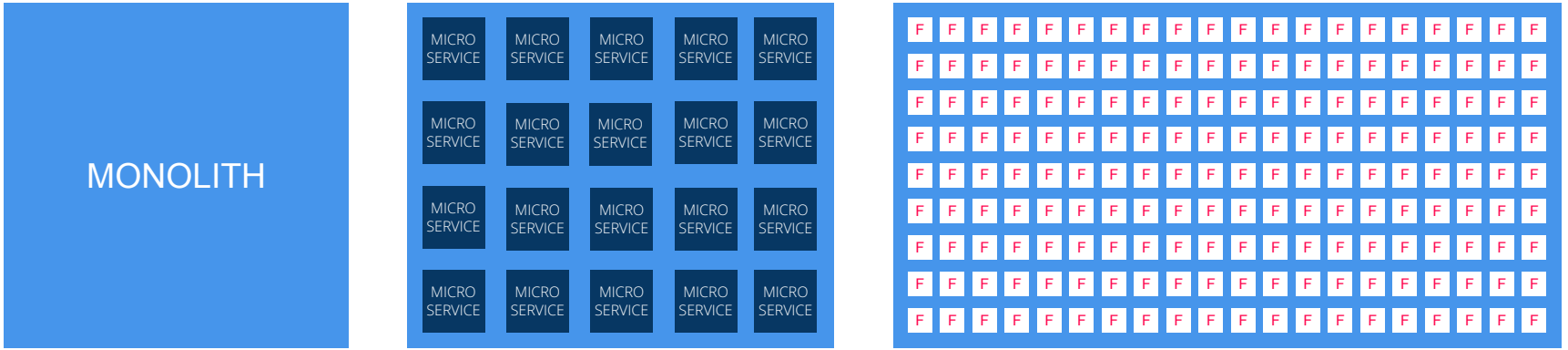
QUARKUS

Subatomic Supersonic KAFKA!

Clement Escoffier
Sr. Principal Software Engineer
Red Hat

@clementplop

MONOLITH - - - MICROSERVICE - - - SERVERLESS



1 monolith ≈ 20 microservices ≈ 200 functions

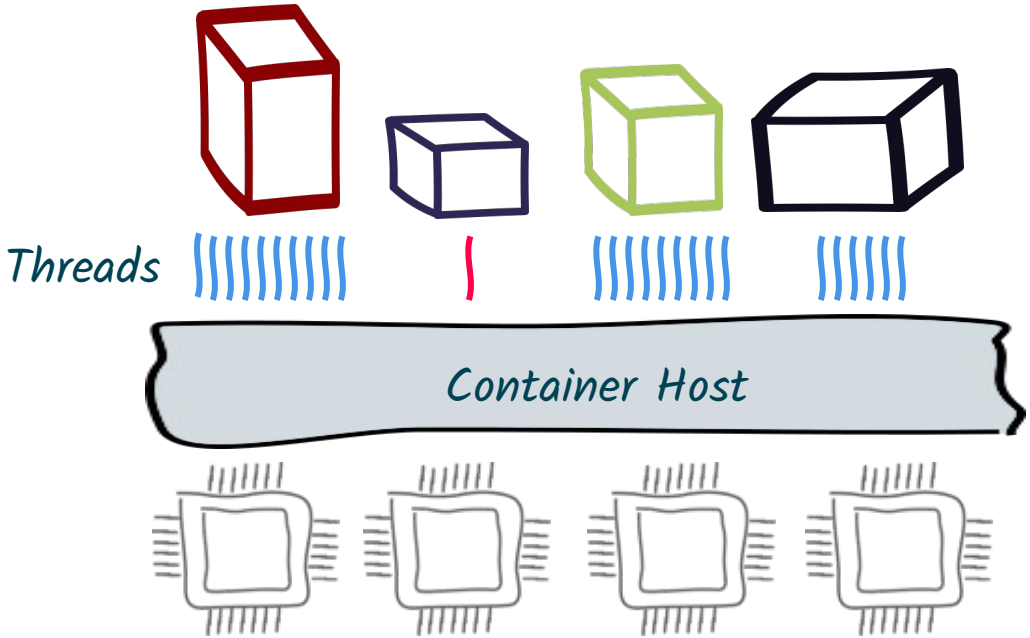
Scale to 1 vs scale from 0 to n

Startup time

Distributed Systems all the way!

The hidden truth of containers

Containers are about **sharing** and **deployment density**



costs memory

cost lots of memory

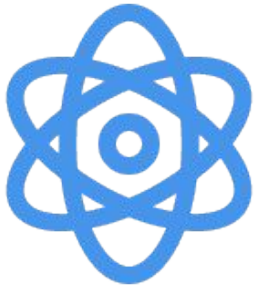
cost CPU cycles

\+ quotas = BOOM

Quarkus!

A stack to write Java apps

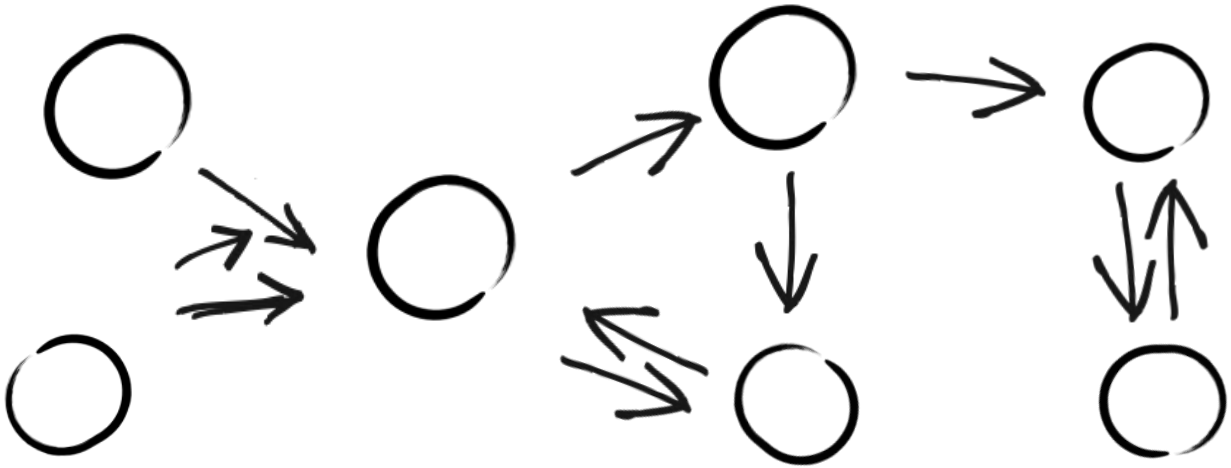
Cloud Native



Microservices

Serverless

A stack to build



Distributed Systems

HTTP

Cloud

EIP

Workflow

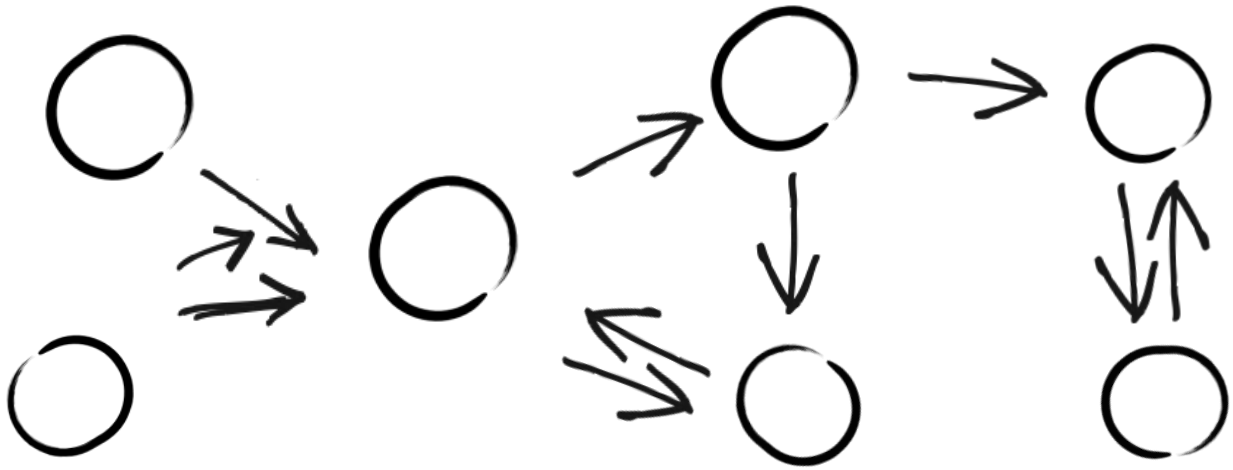
Kafka

A stack to build

REST

AMQP

Database



GraphQL

Security

LRA

gRPC

Distributed Systems

CDC

CLI

Container

Monolith

Microservices

Functions

HTTP

Cloud

EIP

Workflow

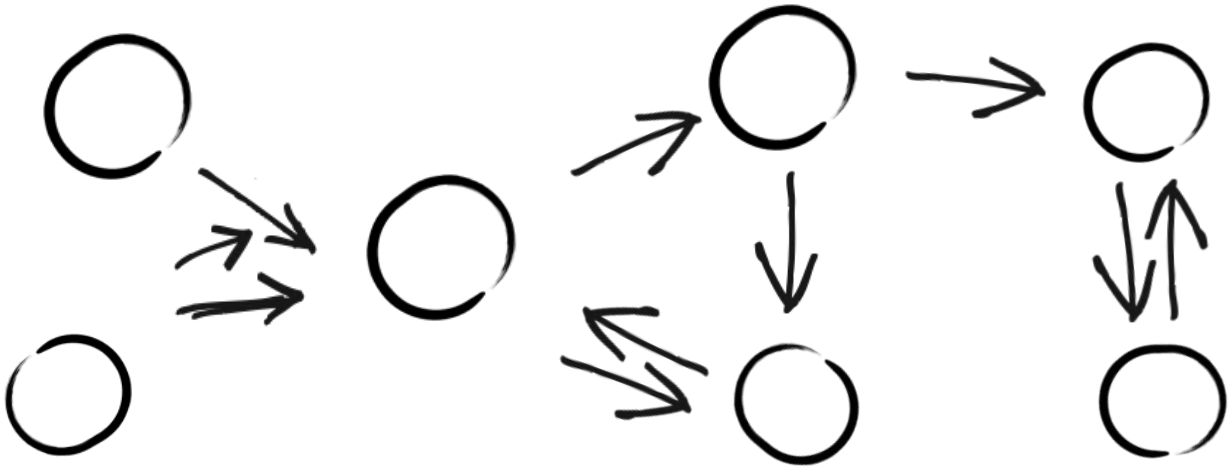
Kafka

A stack to build

REST

AMQP

Database



GraphQL

Security

LRA

gRPC

Distributed Systems

CDC

CLI

Container

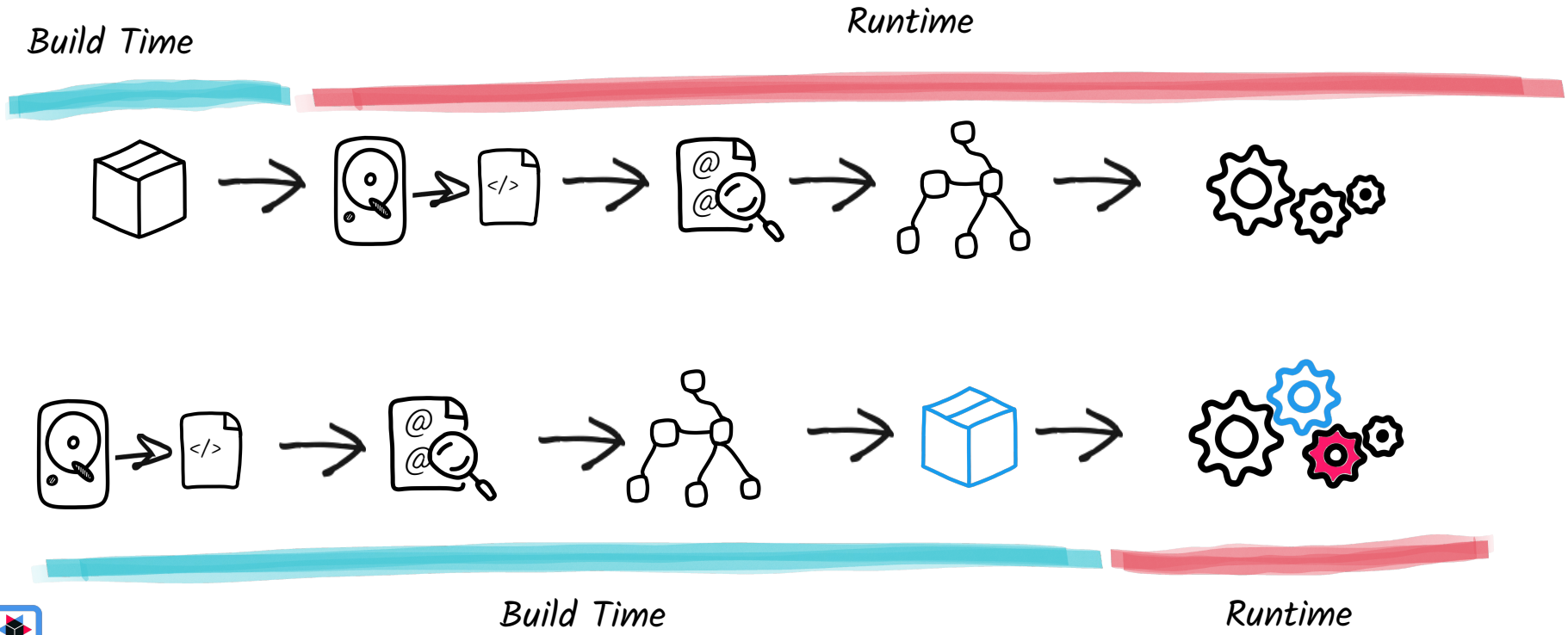
Monolith

Microservices

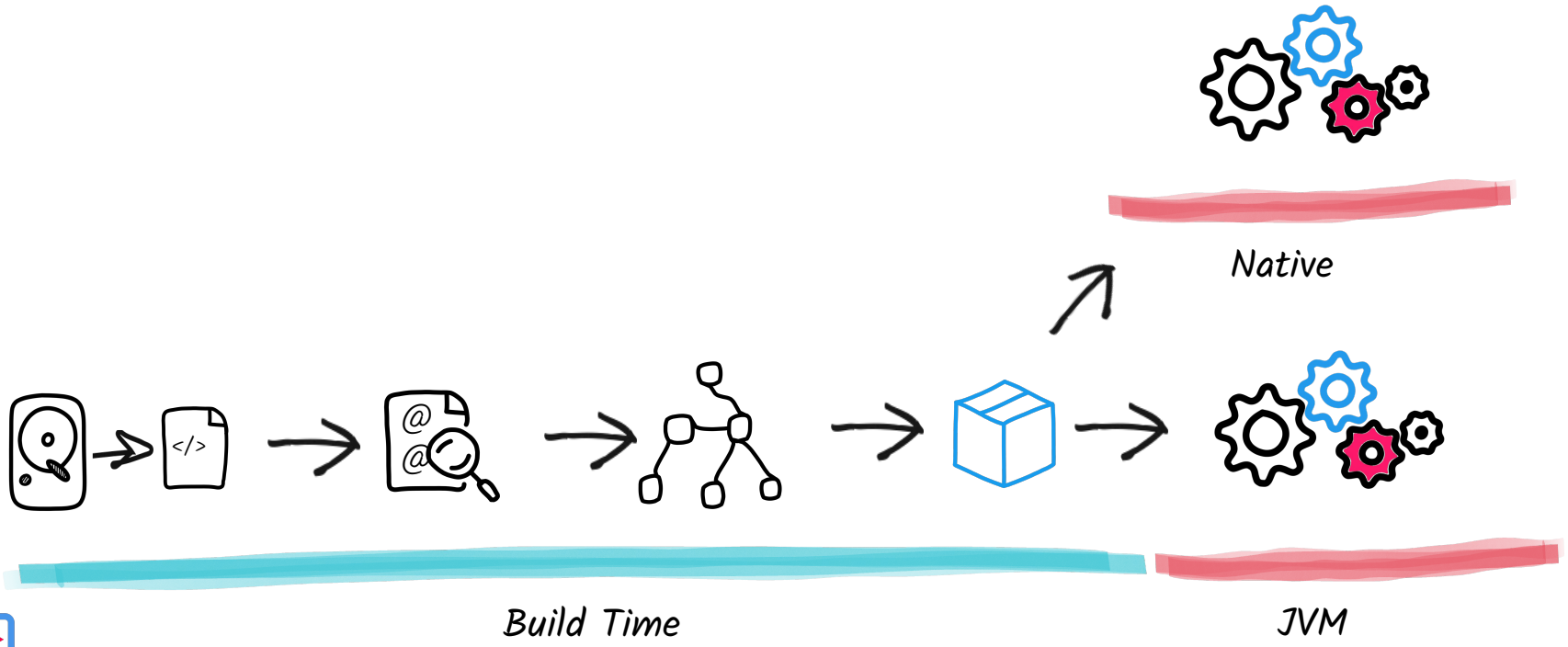
Functions

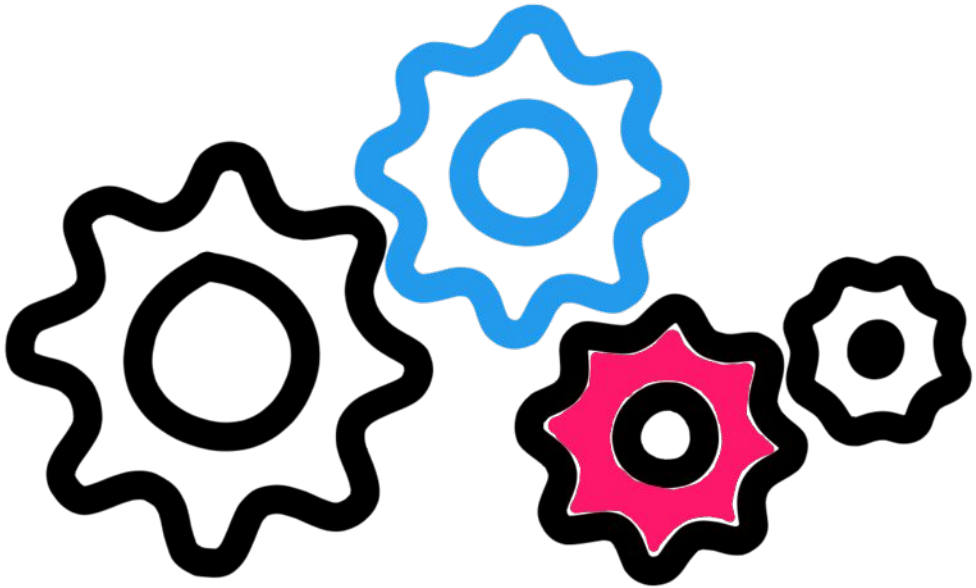
Quarkus = Build Time + Reactive

The Traditional vs. Quarkus Ways

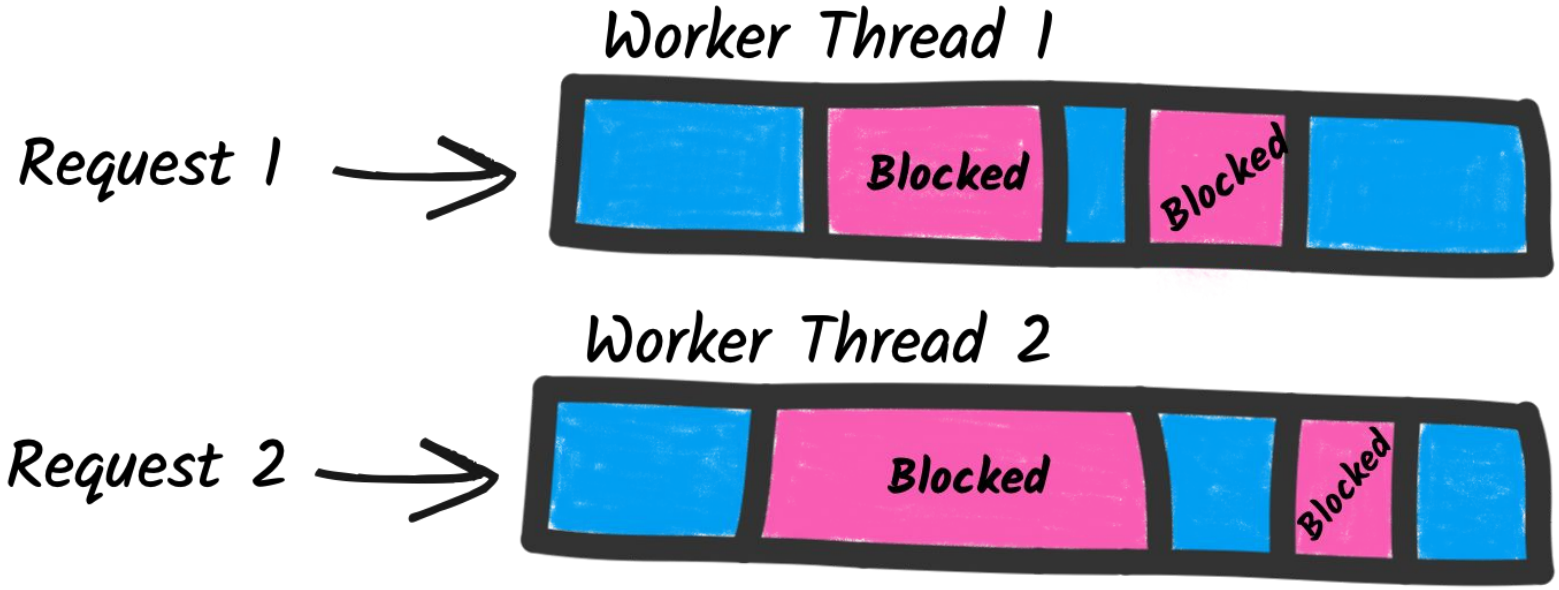


The Quarkus Way enables native compilation

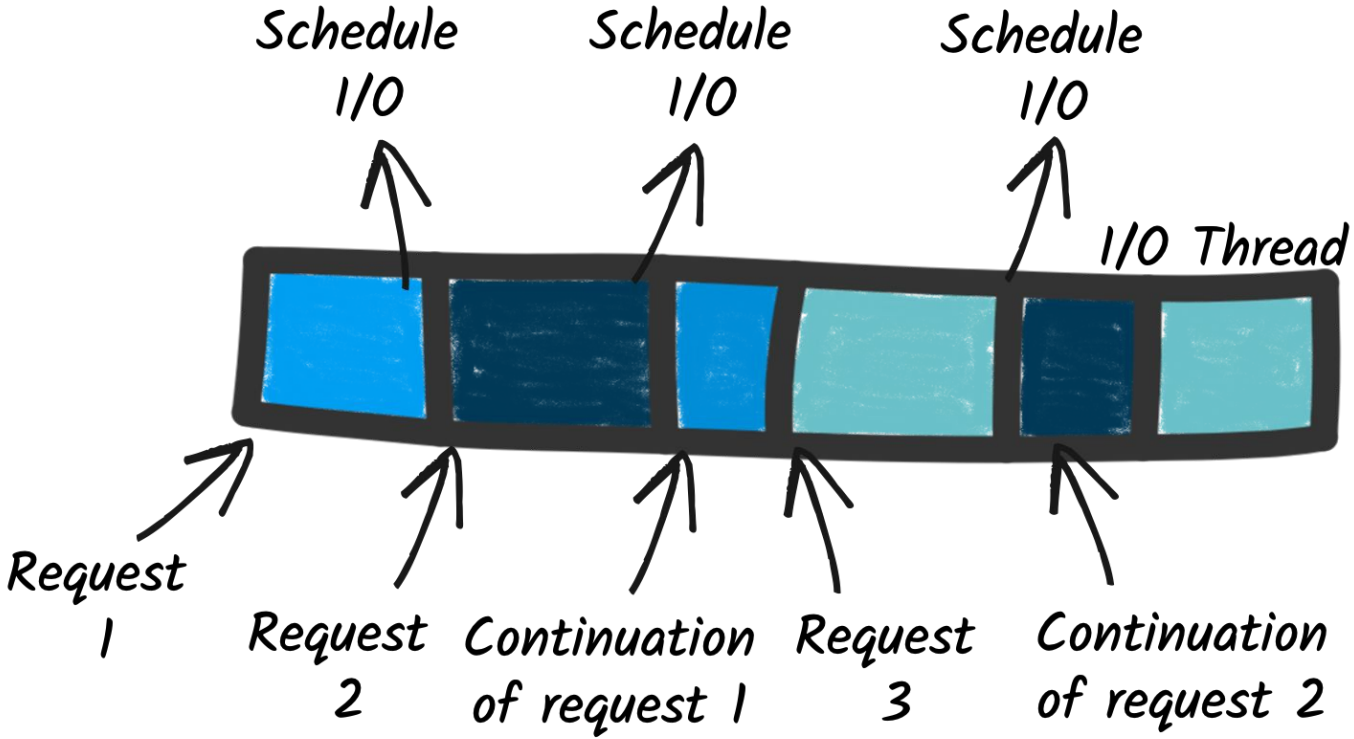




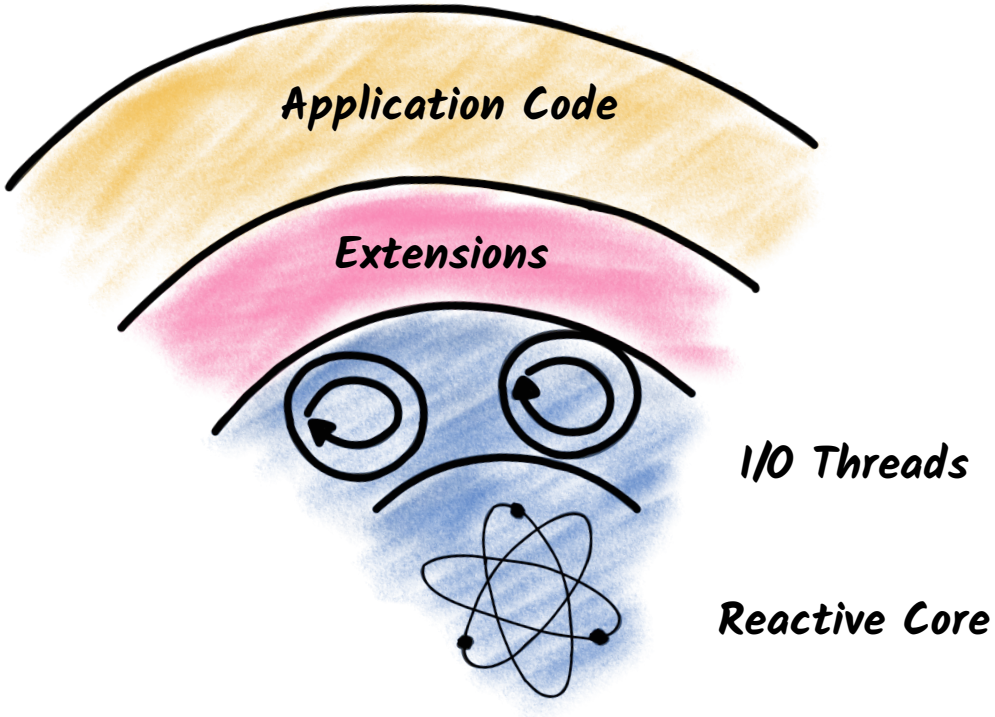
Imperative Execution Model



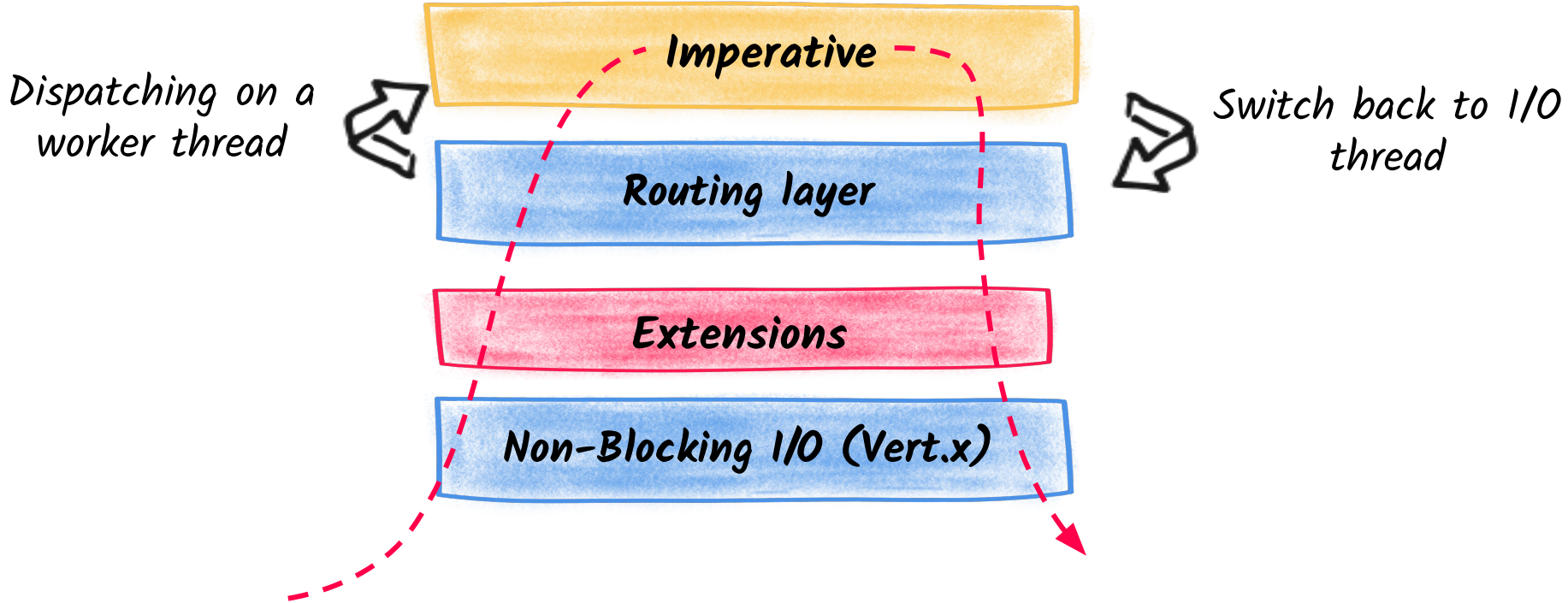
Reactive Execution Model



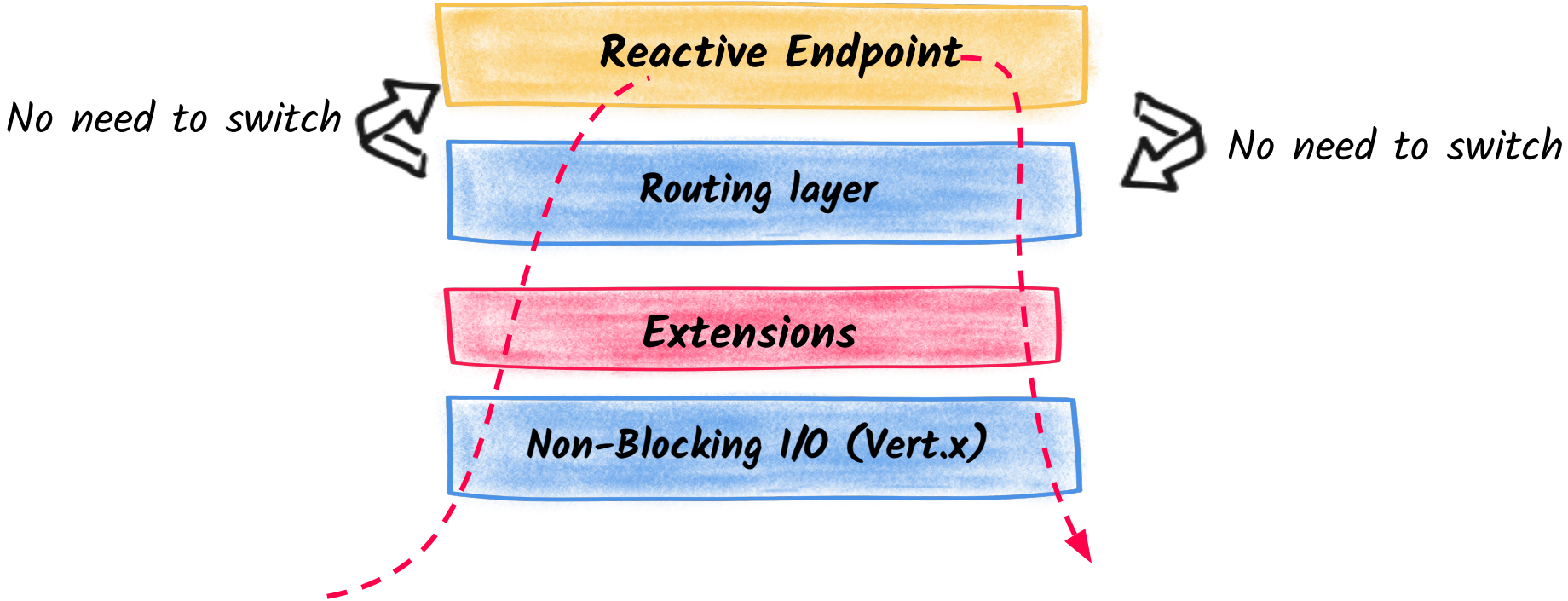
Reactive Core



Quarkus *smart* Routing

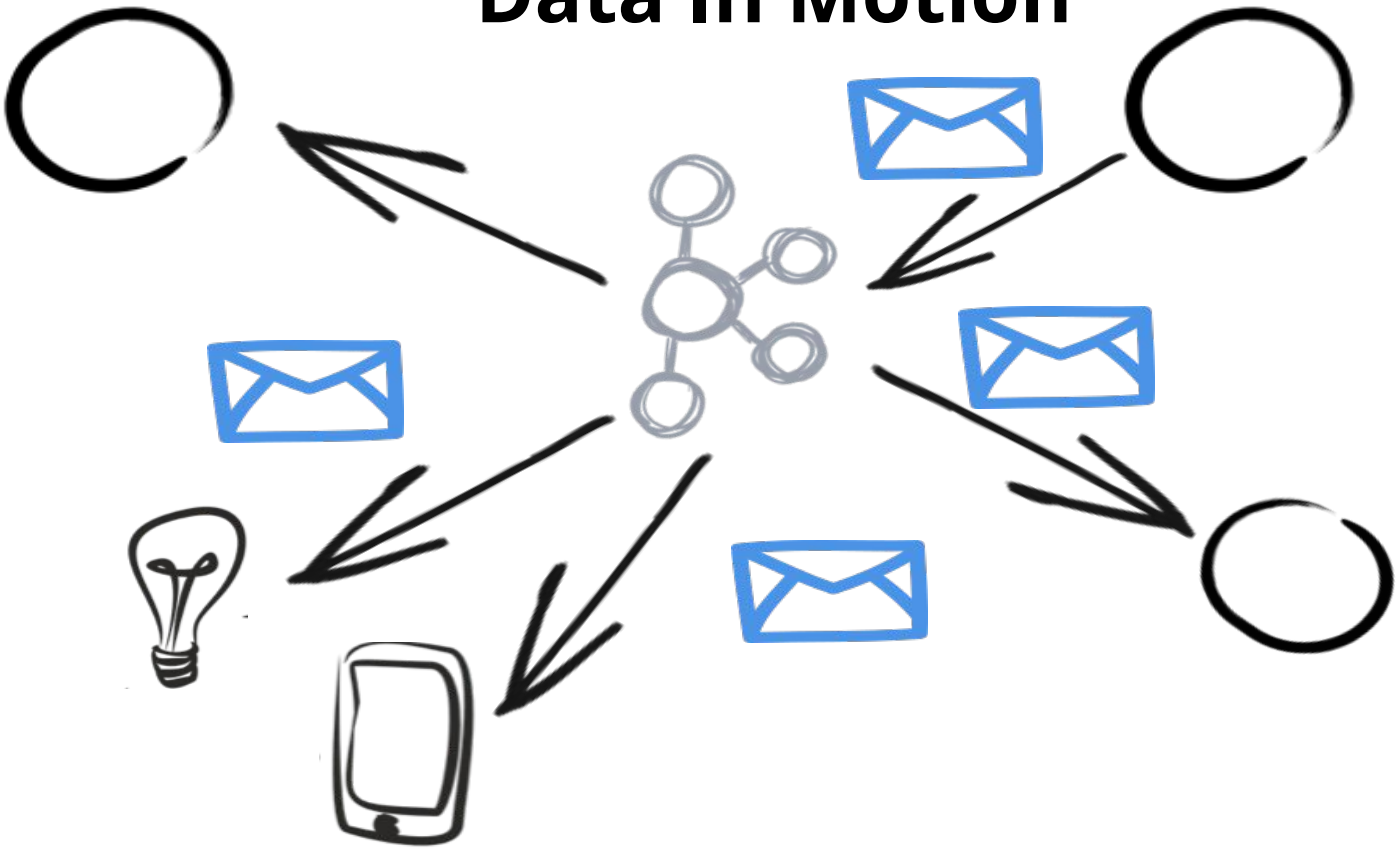


Quarkus *smart* Routing



Kafka...

Data In Motion

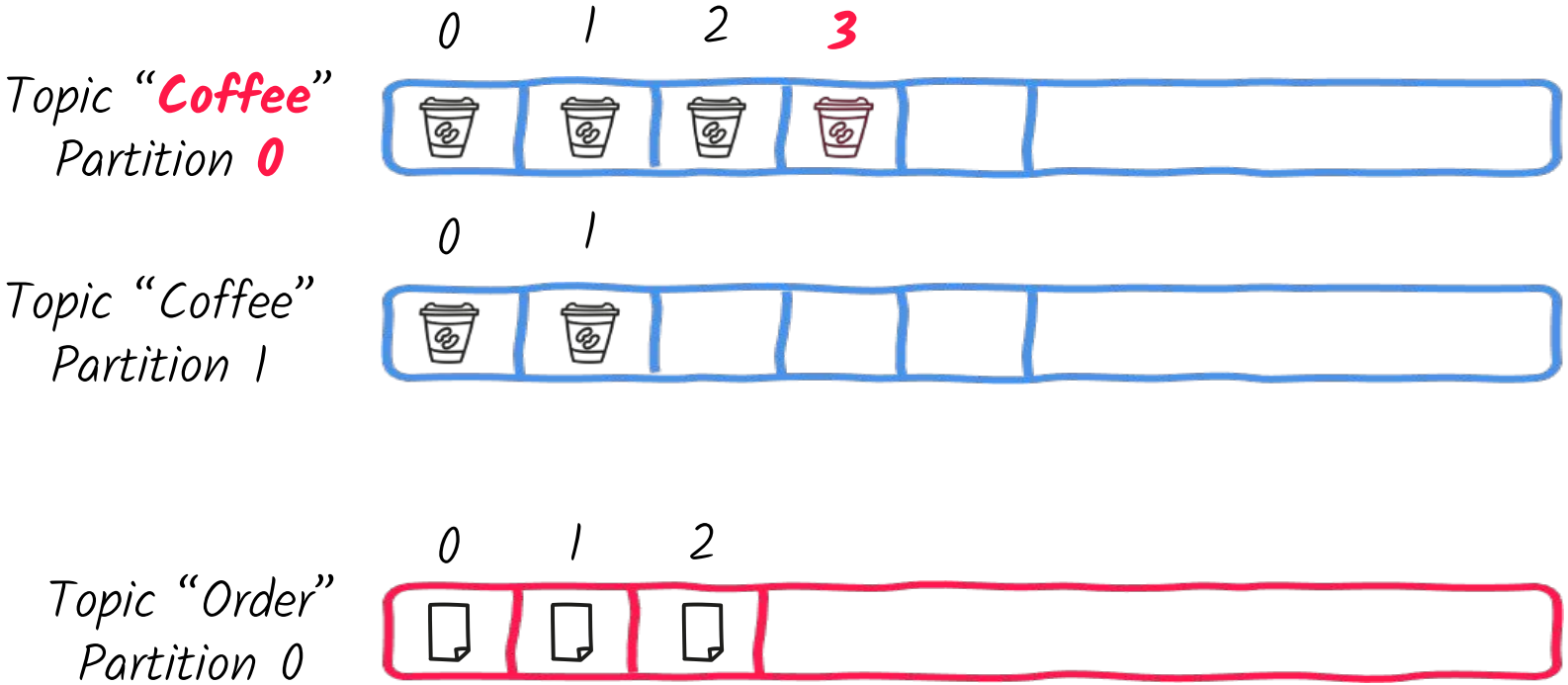


At the center of Kafka, there are records

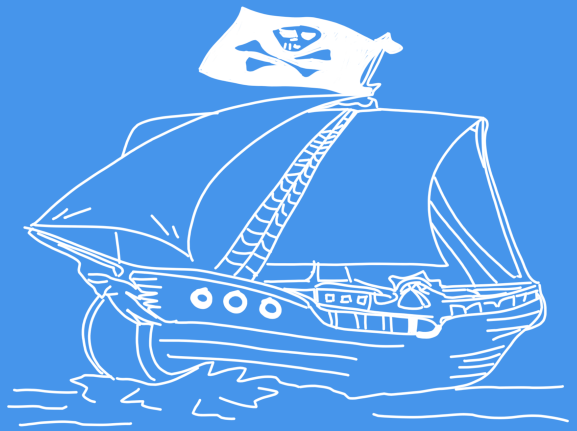
```
// Producer:

ProducerRecord<String, String> record =
  new ProducerRecord<>("topic", "clement", "some data");

ProducerRecord<String, String> record2 =
  new ProducerRecord<>("topic", "value");

// ------------------------------------------------

// Consumer:

Record<String, String> record3 = …;
String key = records3.key();
String value = records3.value();
```

Records are written to partitions

Partitions are grouped into topics



Kafka is simple...

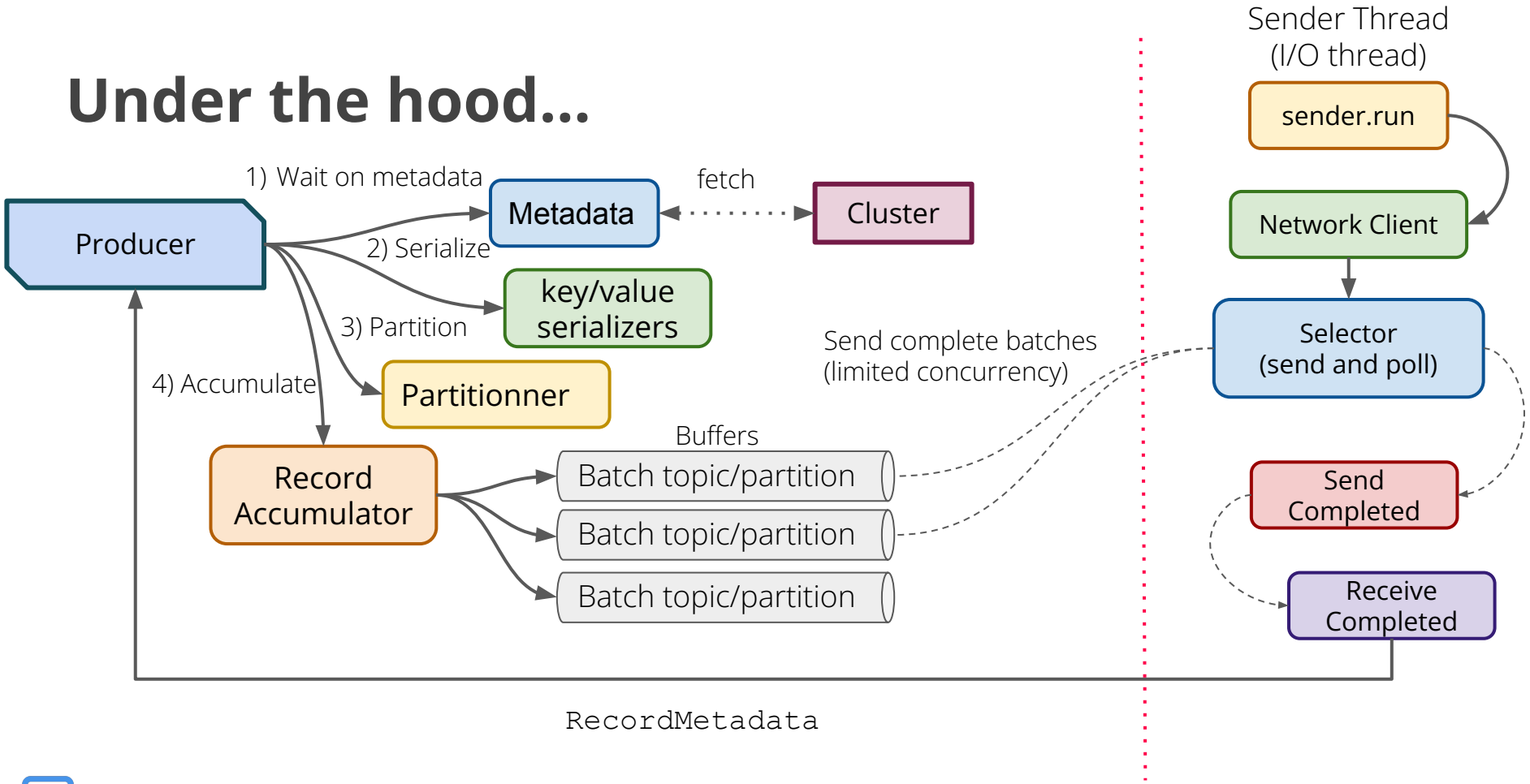


Producing records

```
public KafkaProducer<String, Person> getProducer() {
    Map<String, Object> config = new HashMap<>();
    config.put(BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
    config.put(KEY_SERIALIZER_CLASS_CONFIG,
                    StringSerializer.class.getName());
    config.put(VALUE_SERIALIZER_CLASS_CONFIG,
                    ObjectMapperSerializer.class.getName());
    return new KafkaProducer<>(config);
}
```

```
Future<RecordMetadata> future = producer.send(new
ProducerRecord<>("coffee", "01234", "espresso"));
```

Under the hood...



Consuming Records

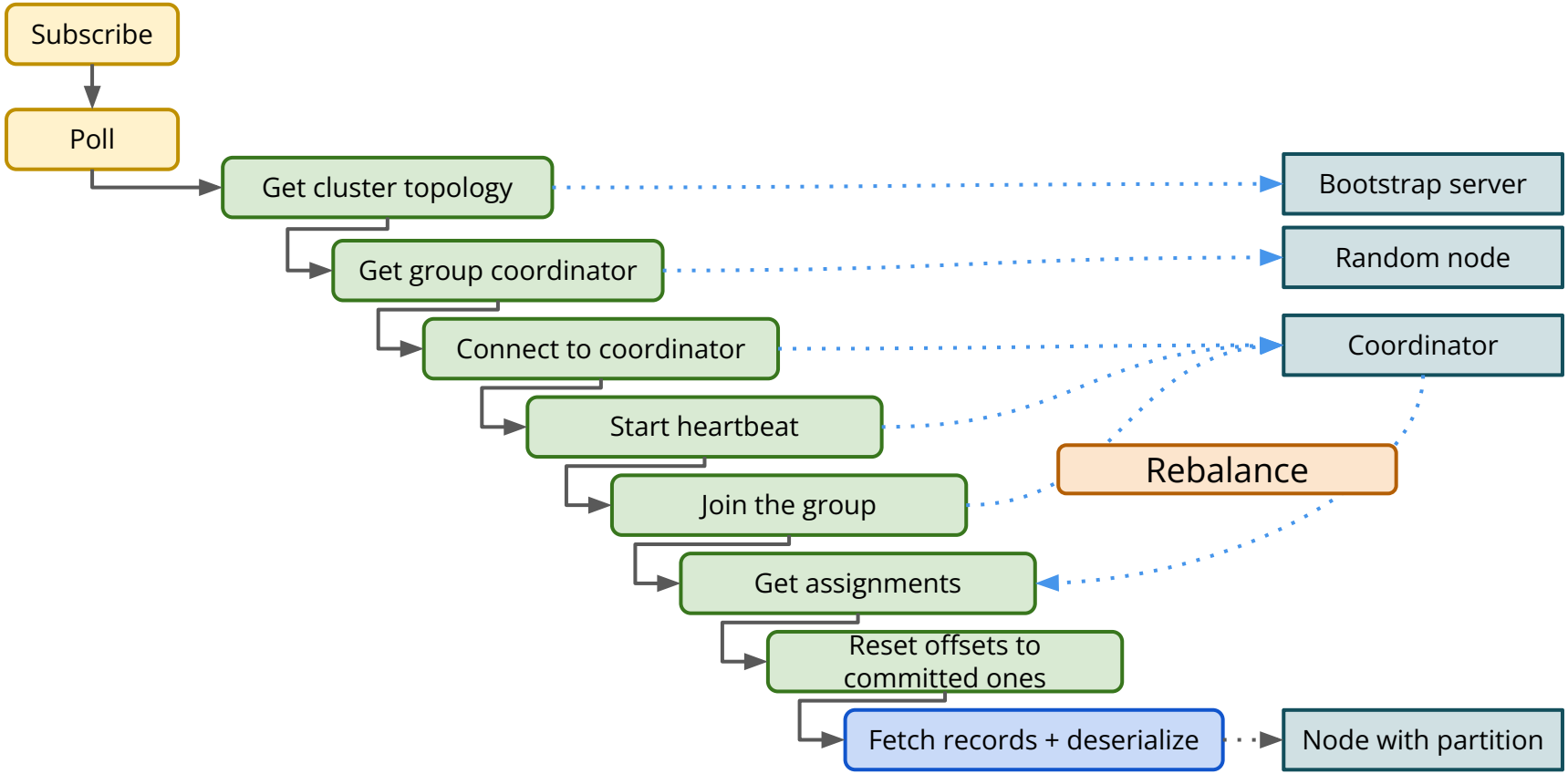
```
Map<String, Object> cfg = Map.of(/* … */);
KafkaConsumer<Key, Value> consumer = new KafkaConsumer<>(cfg);
consumer.subscribe(List.of("topic"));
while(true) {
    var records = consumer.poll(Duration.ofSeconds(1));
    for(ConsumerRecord<Key, Value> record : records) {
        // process record (write to DB, write to Kafka topic, etc.)
        // commit offset of processed records
    }
}
```

Consuming Records

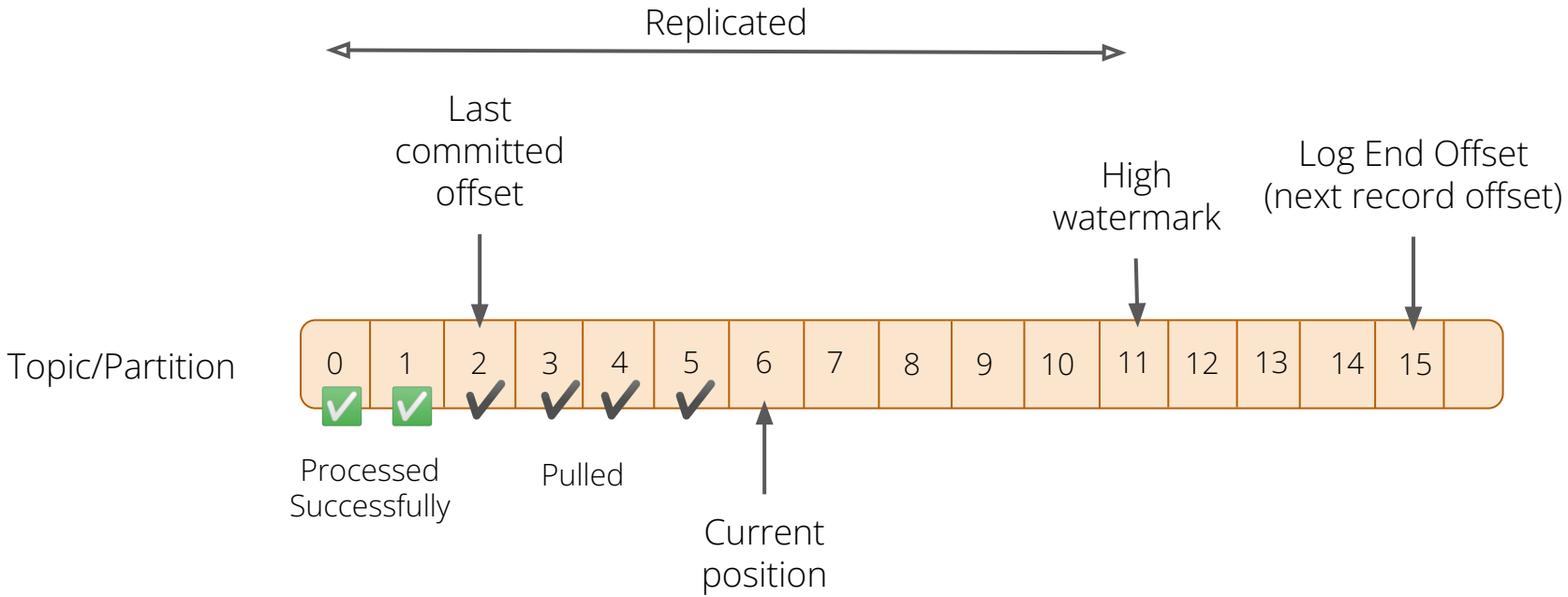
```
Map<String, Object> cfg = Map.of(/* … */);
KafkaConsumer<Key, Value> consumer = new KafkaConsumer<>(cfg);
consumer.subscribe(List.of("topic"));
while(true) {
    var records = consumer.poll(Duration.ofSeconds(1));
    for(ConsumerRecord<Key, Value> record : records) {
        // process record (write to DB, write to Kafka topic, etc.)
        // commit offset of processed records
    }
}
```

Consumer heartbeat **Offset commit** **Error handling** **Backpressure pause/resume** **Consumer rebalance**

Because poll is more than polling

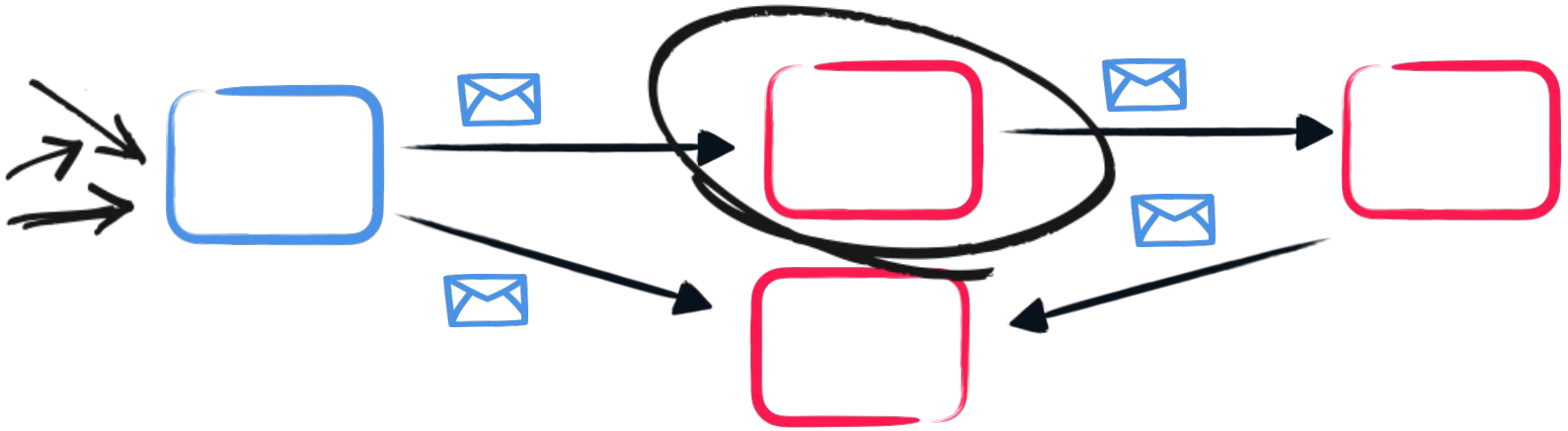


Offsets Management and Commit

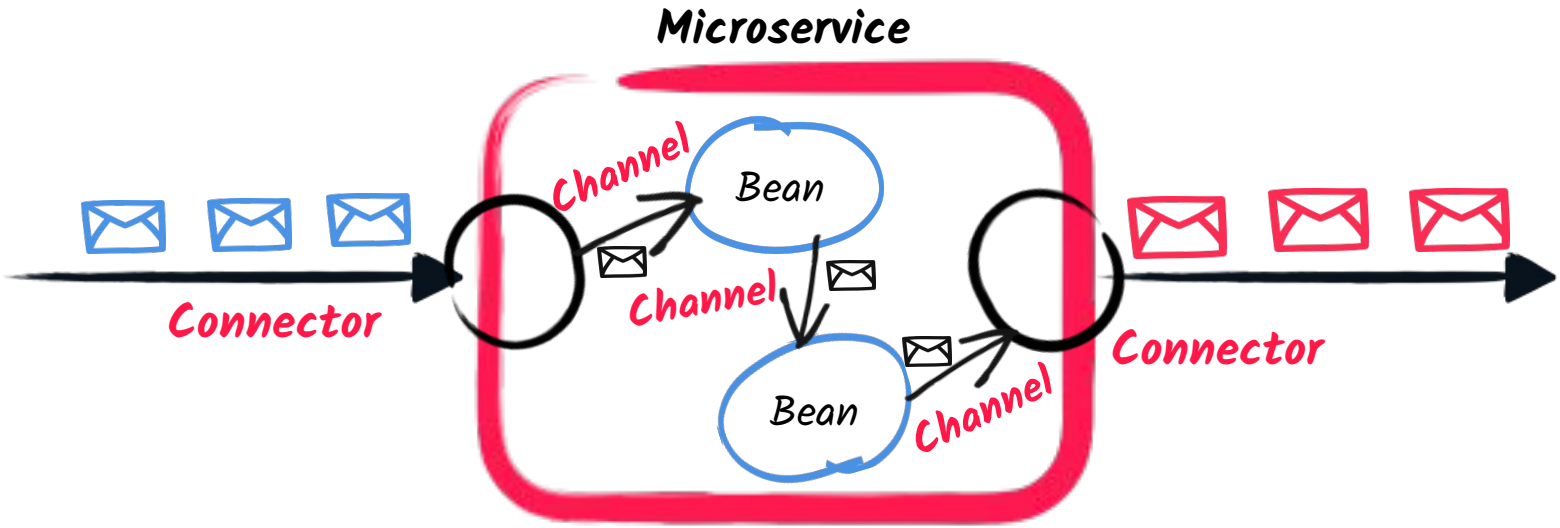


Integrating Kafka In Quarkus

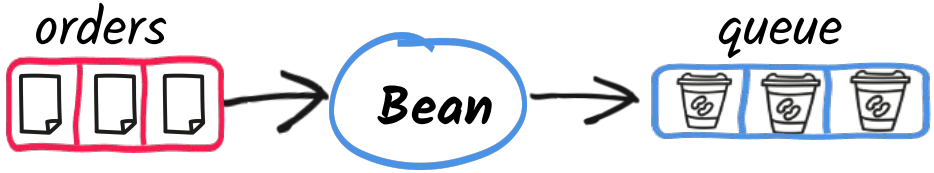
Reactive Messaging: Event-Driven Architectures



Event-Driven Microservices

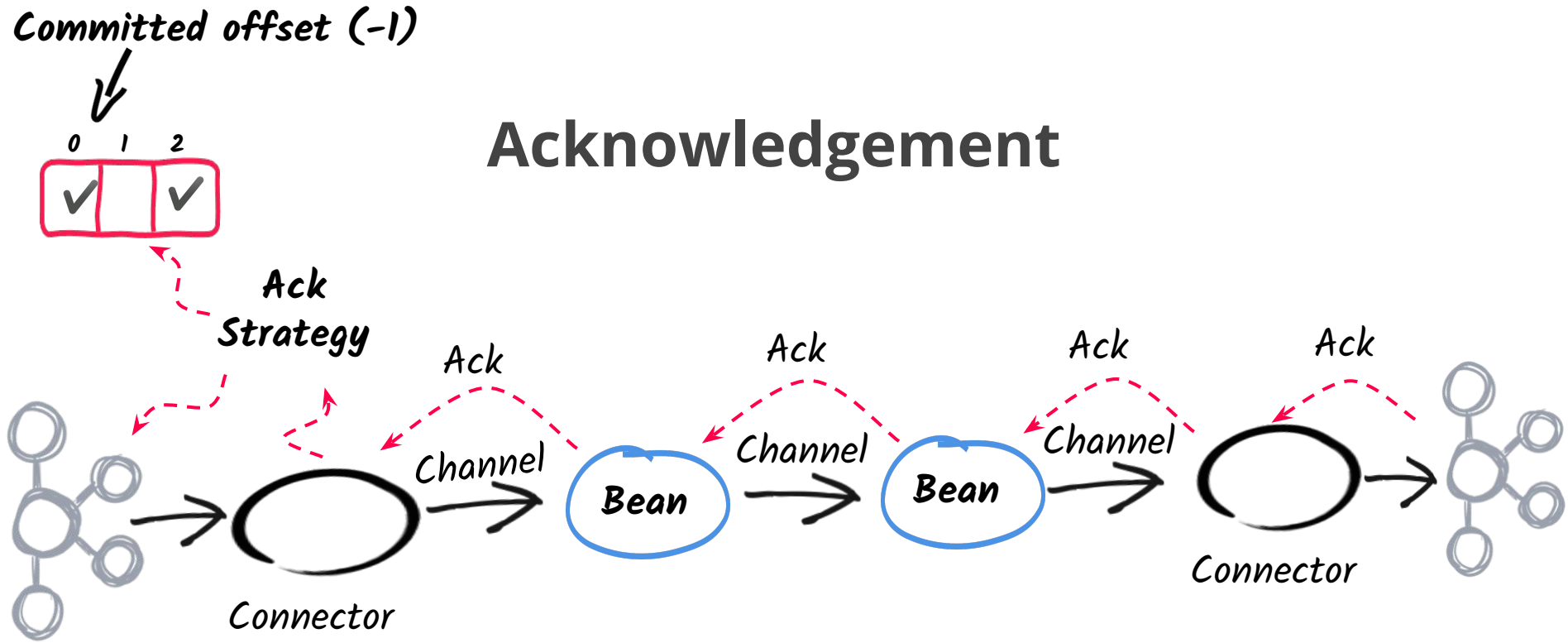


Event-Driven Bean

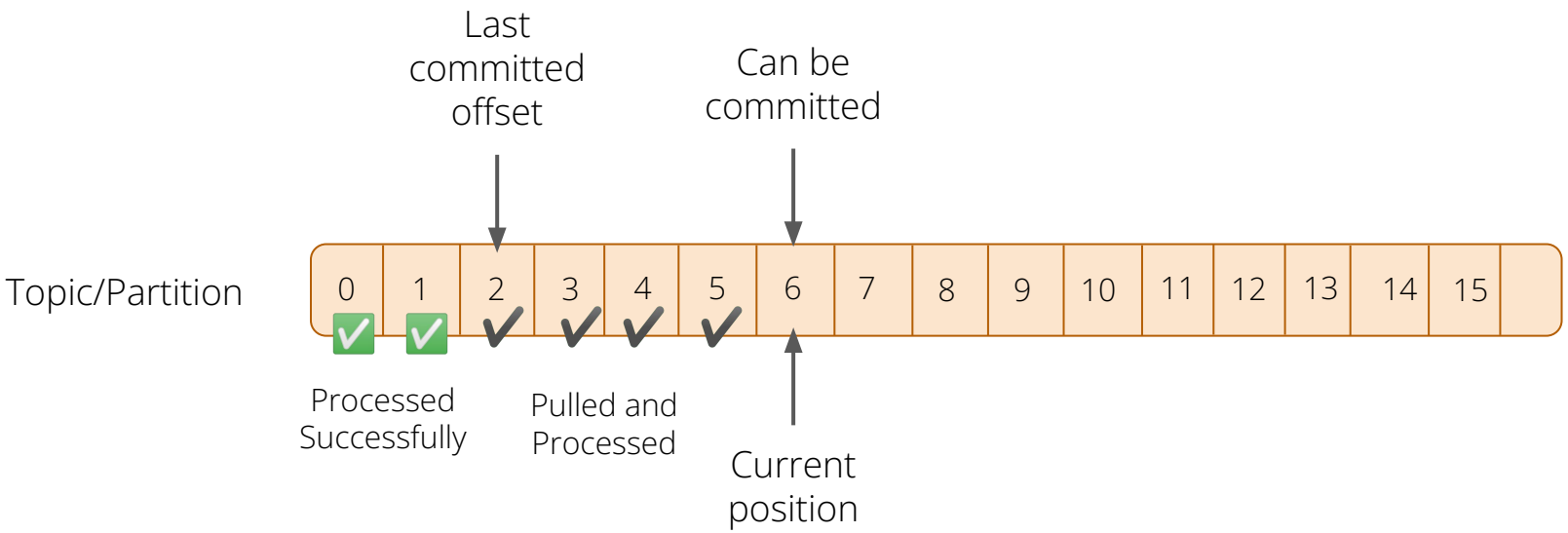



```
@Incoming("orders")
@Outgoing("queue")
@Blocking
public Beverage process(Order order) {
  return process(order);
}
```

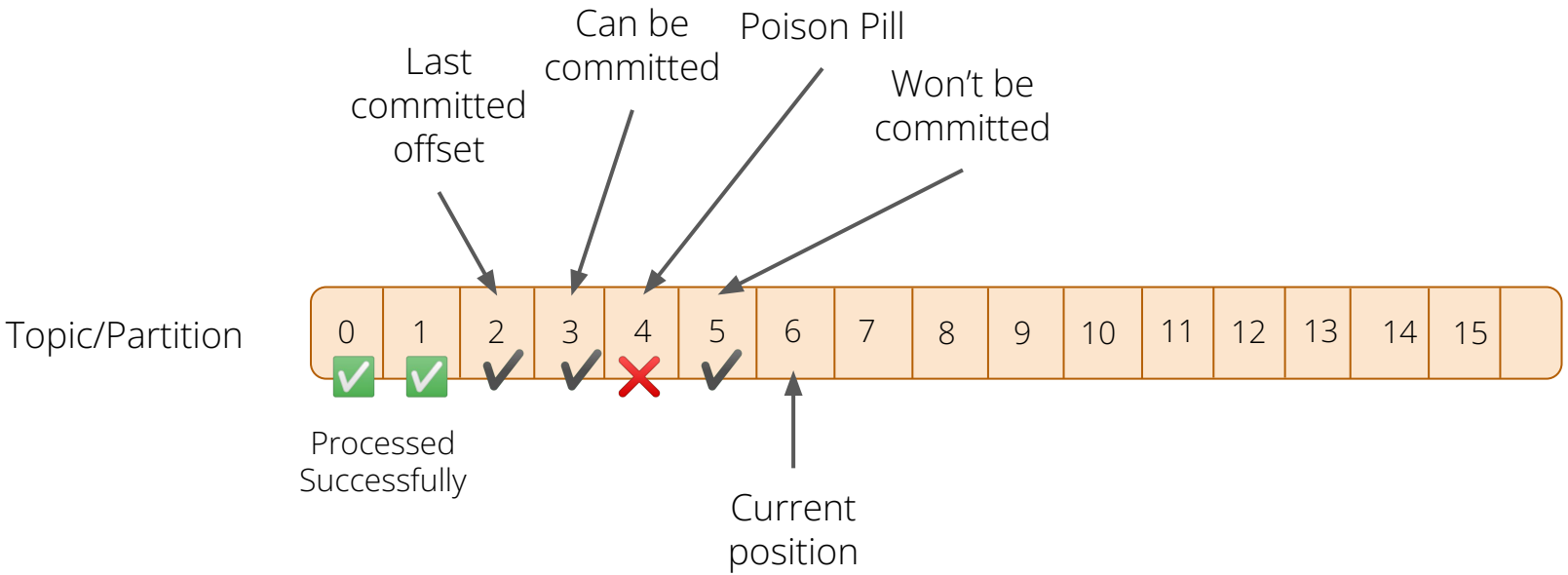
Acknowledgement



Throttled Commit Strategy



Throttled Commit Strategy

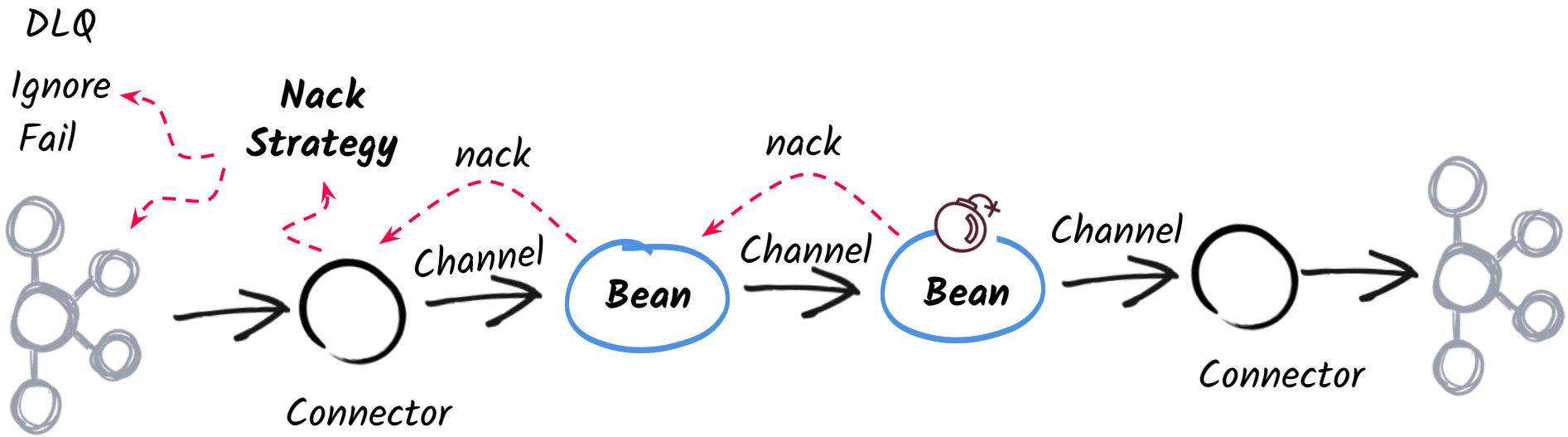


Throttled Commit Strategy - Poison Pill

What happens when a message cannot be processed?

- If the message associated to a polled record is not acknowledged within 60s (configurable), the application is considered **unhealthy**.
- You will never be able to commit any more messages on that partition
- We will also stop polling (as it would lead to an OOM)

Negative-Acknowledgement (nack)



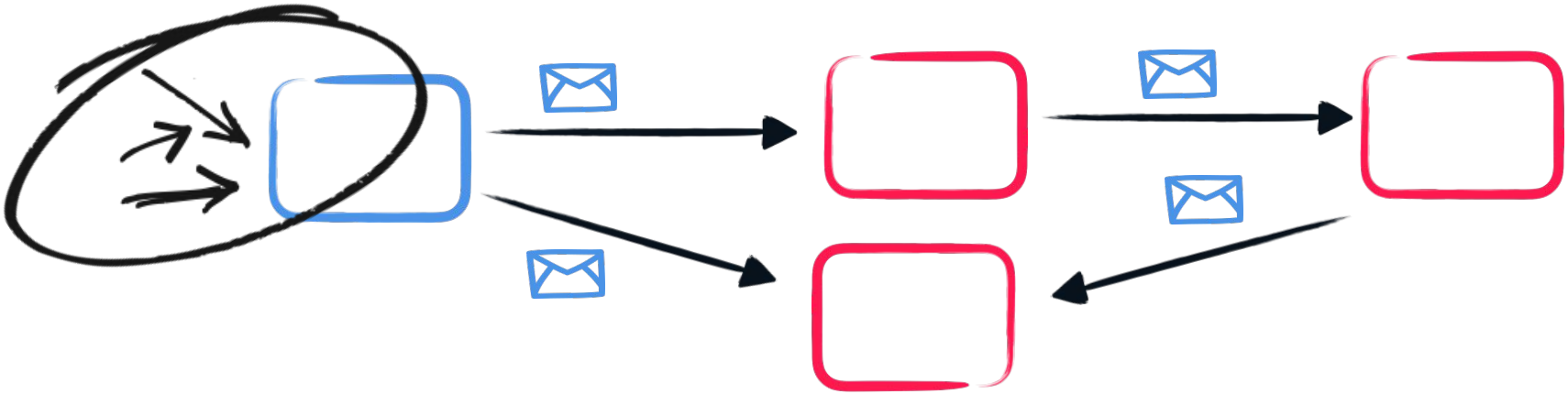
What about retrying

Retrying processing is possible using **@Retry**

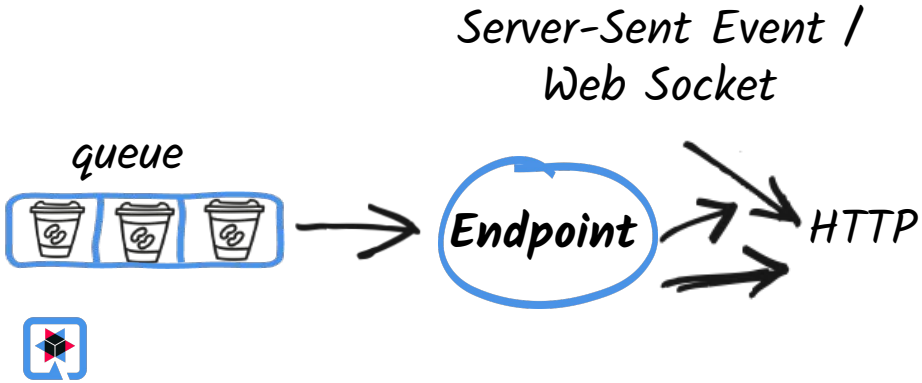
- This does not retry writing to Kafka, only the processing
- You can also use **@Fallback**, **@Timeout** or **@CircuitBreaker**

```
@Incoming("orders")
@Outgoing("queue")
@Retry
Beverage process(Order o) {
  // …
}
```

Bridging HTTP and Messaging

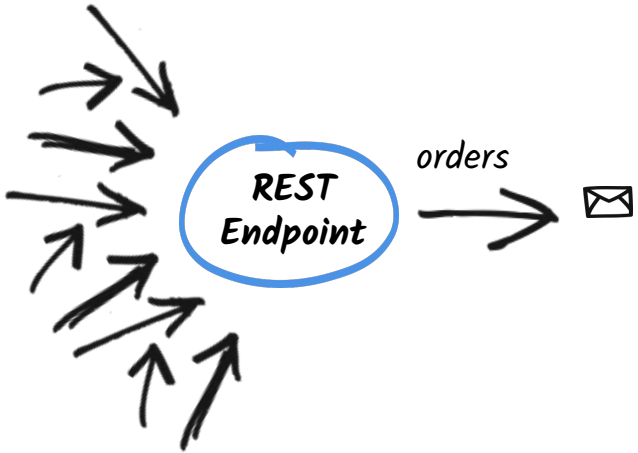


Emitter / Channel




```java
@Channel("orders")
Emitter<Order> orders;
@Channel("beverages")
Multi<Beverage> queue;

@POST
Order kafka(Order o) {
    orders.send(o);
}
@GET
@Produces(SERVER_SENT_EVENTS)
Multi<Beverage> queue() {
    return queue;
}
```

Overflow

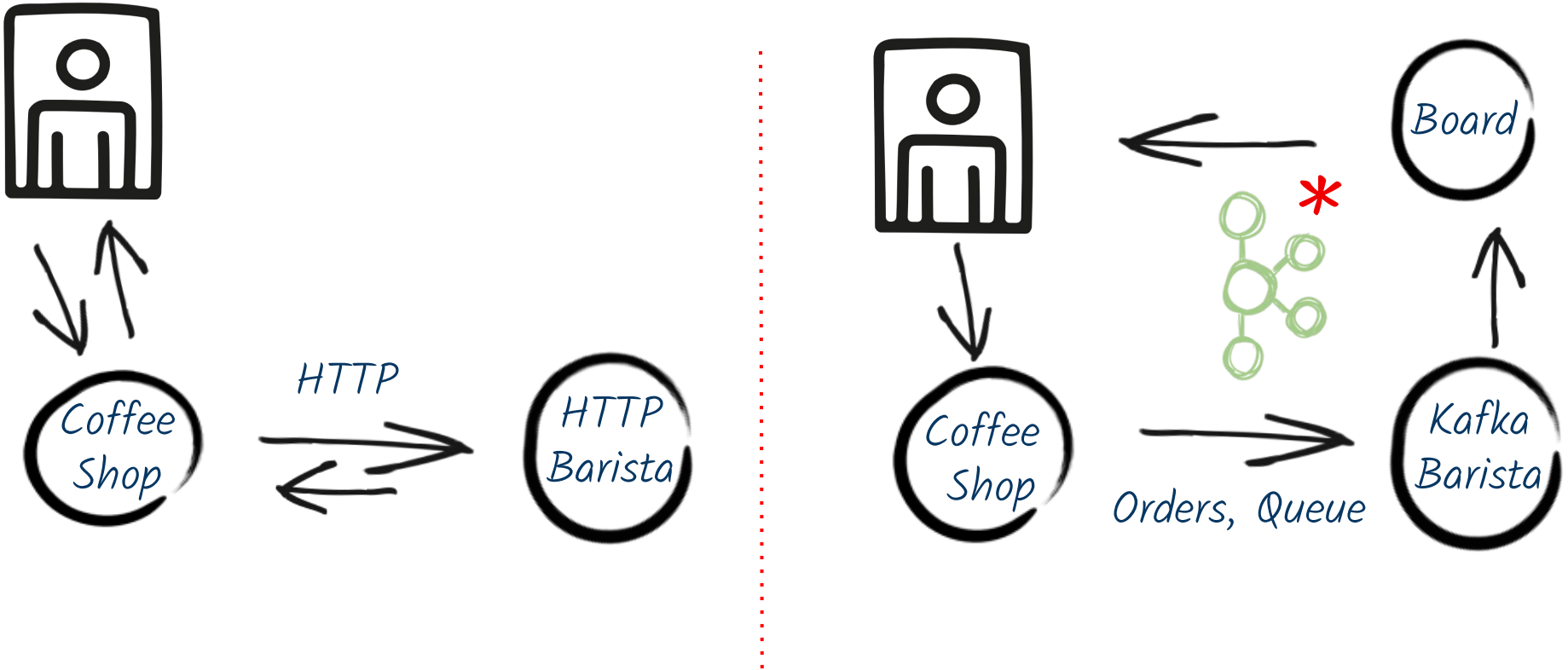



```
@Path("/orders")
public class OrderResource {

  @Inject
  @Channel("orders")
  @OnOverflow(value = OnOverflow.Strategy.BUFFER,
              bufferSize = 1000)
  Emitter<Order> emitter;

  @GET
  public CompletionStage<Void> newOrder(Order order)  {
    return emitter.send(order);
  }


}
```

Time for some code!

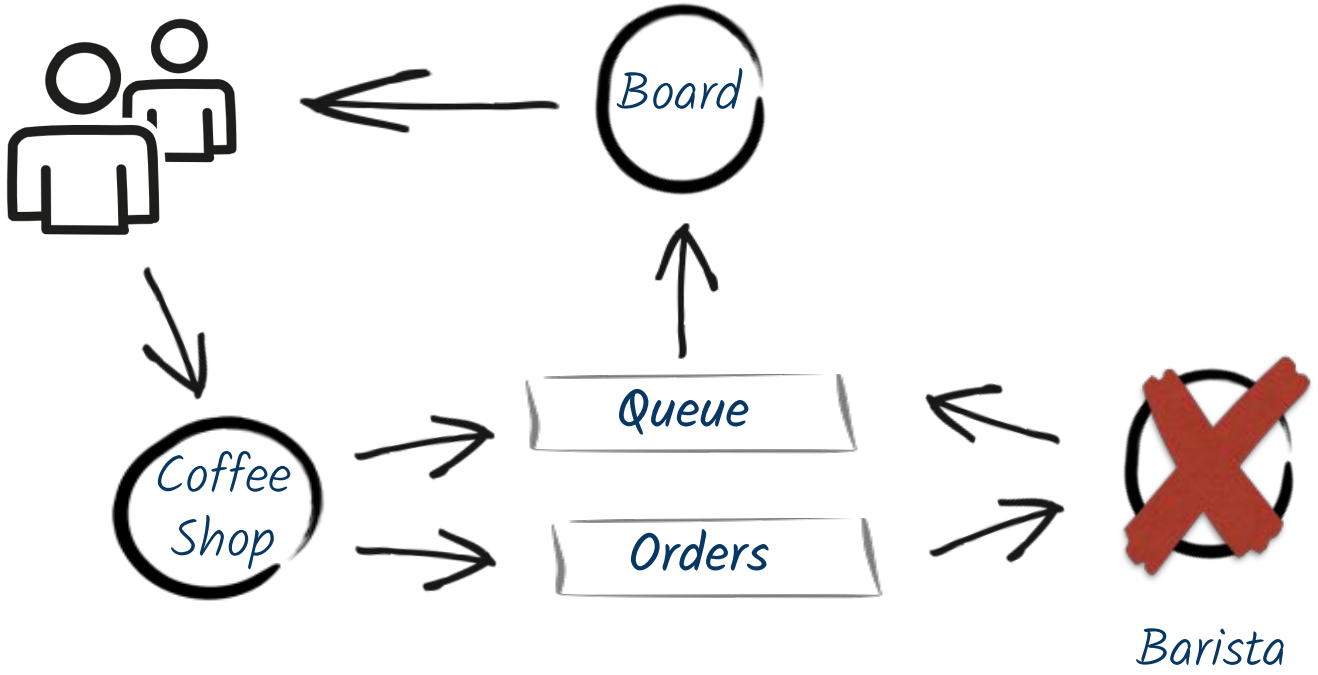
Reactive Coffee Shop

https://github.com/cescoffier/reactive-coffeeshop-demo

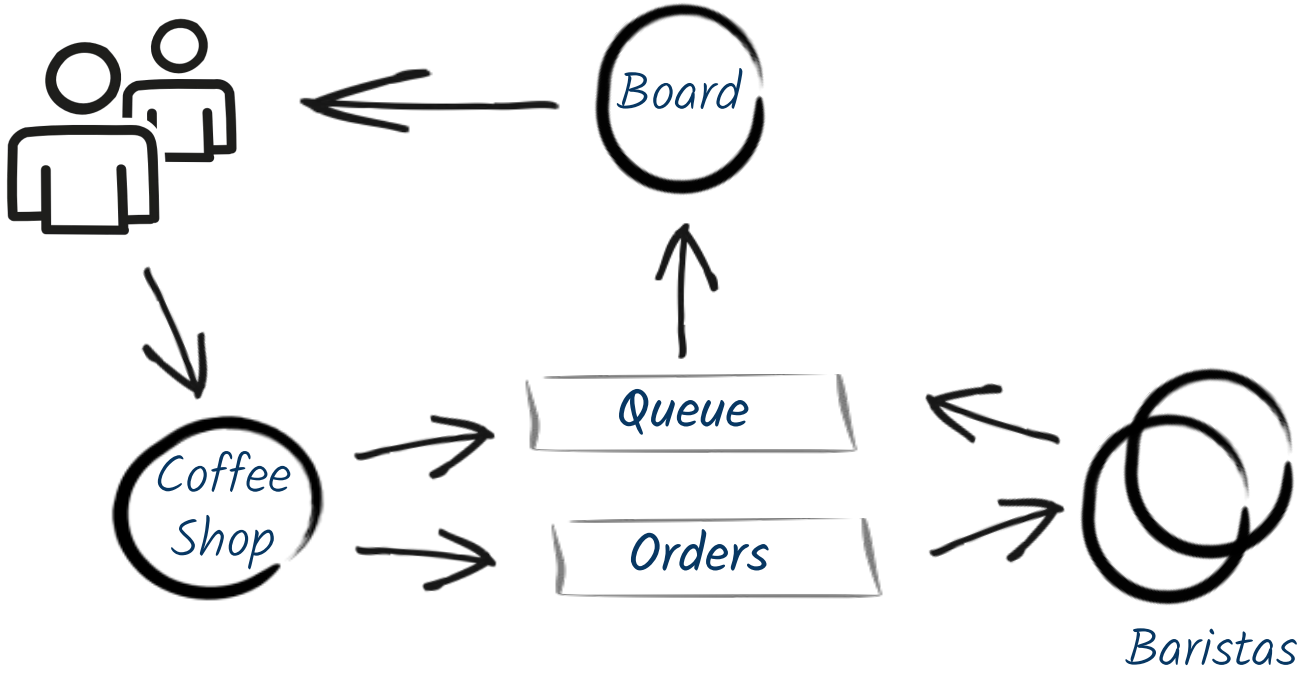


*Also applies to AMQP 1.0, RabbitMQ, Camel... but we want to be "cool"

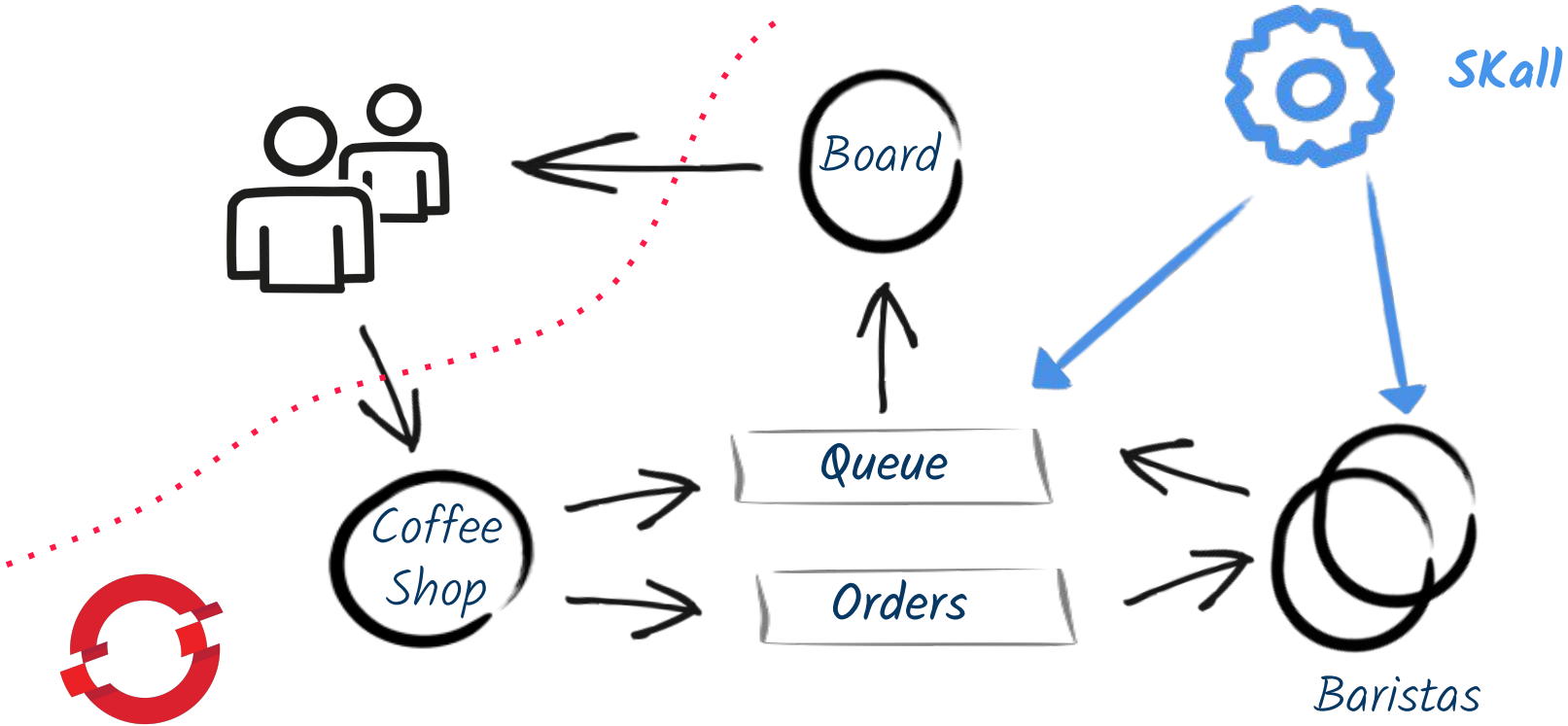
Baristas take breaks



Rush Hour!

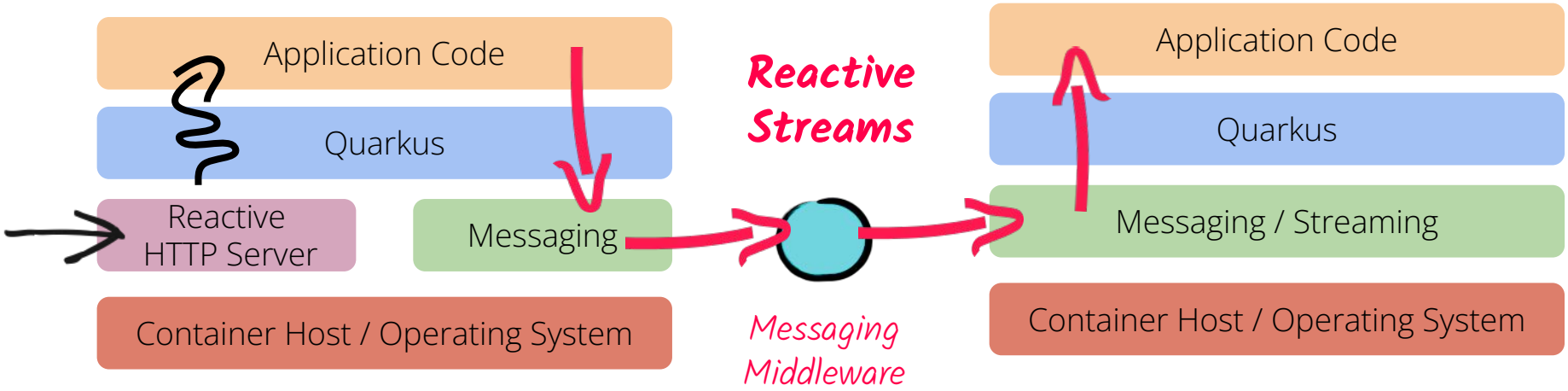


Cloudy & Stormy Days

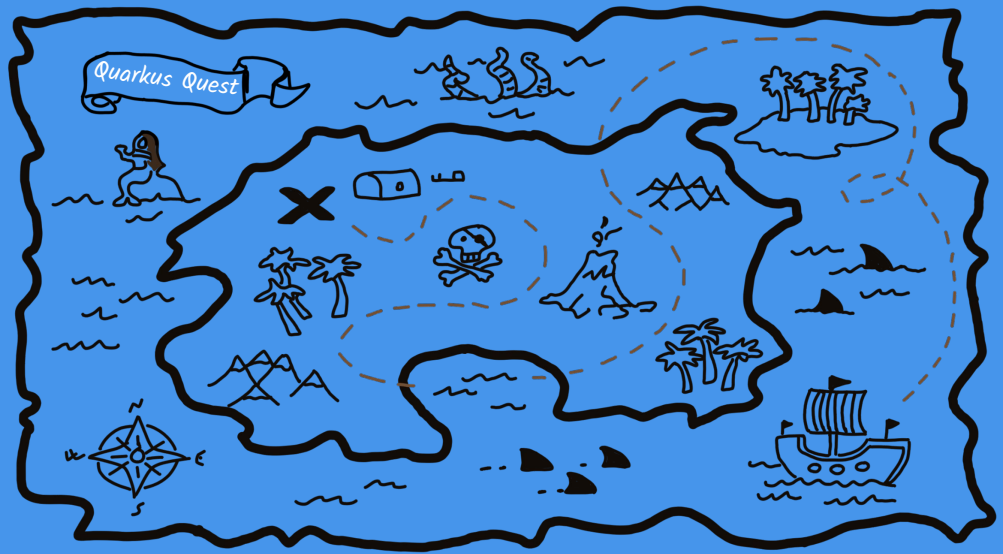


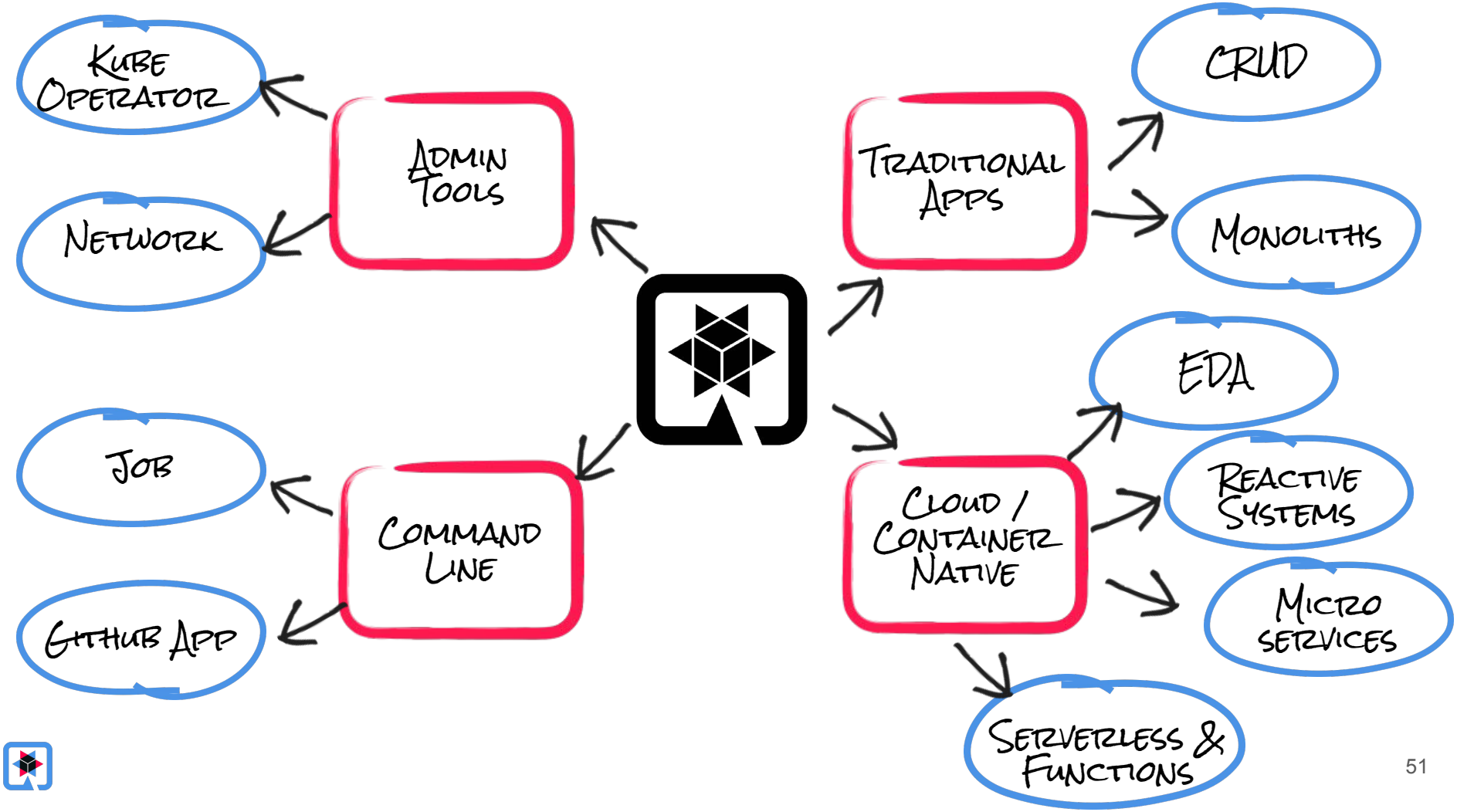
OPENSHIFT

What About Back-Pressure?

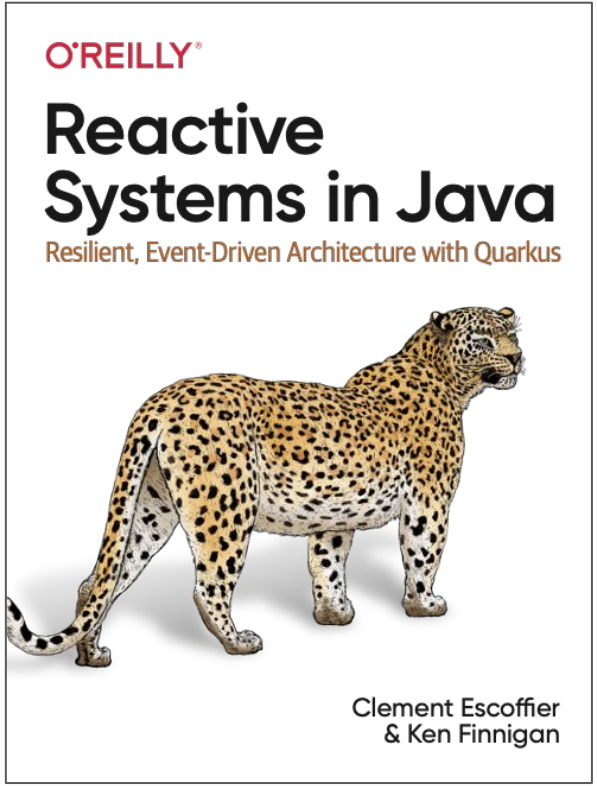


The Famous Last Words




Kube Operator
Network
Admin Tools
Job
Github App
Command Line
Traditional Apps
CRUD
Monoliths
Cloud / Container Native
EDA
Reactive Systems
Micro services
Serverless & Functions

Going further



Thank you.

QUARKUS

https://quarkus.io

https://quarkusio.zulipchat.com

https://youtube.com/quarkusio

@quarkusio

If you like Quarkus, star it on GitHub!
https://github.com/quarkusio/quarkus