WATERMILL

THE MISSING STANDARD LIBRARY FOR EVENT-DRIVEN APPLICATIONS

ROBERT LASZCZAK

CO-FOUNDER OF three dots labs

ROBERT LASZCZAK

CO-FOUNDER OF three dots labs

CREATOR OF Watermill

A TALE ABOUT WATERMILL...

“HOW I CAN MAKE BUILDING EVENT-DRIVEN SERVICES AS SIMPLE AS HTTP API?”

+7K

GITHUB STARS

+100

CONTRIBUTORS

12

OFFICIALLY SUPPORTED PUB/SUBS

kafka

RabbitMQ

NATS

BoltDB

redis

MySQL™

kafka

RabbitMQ

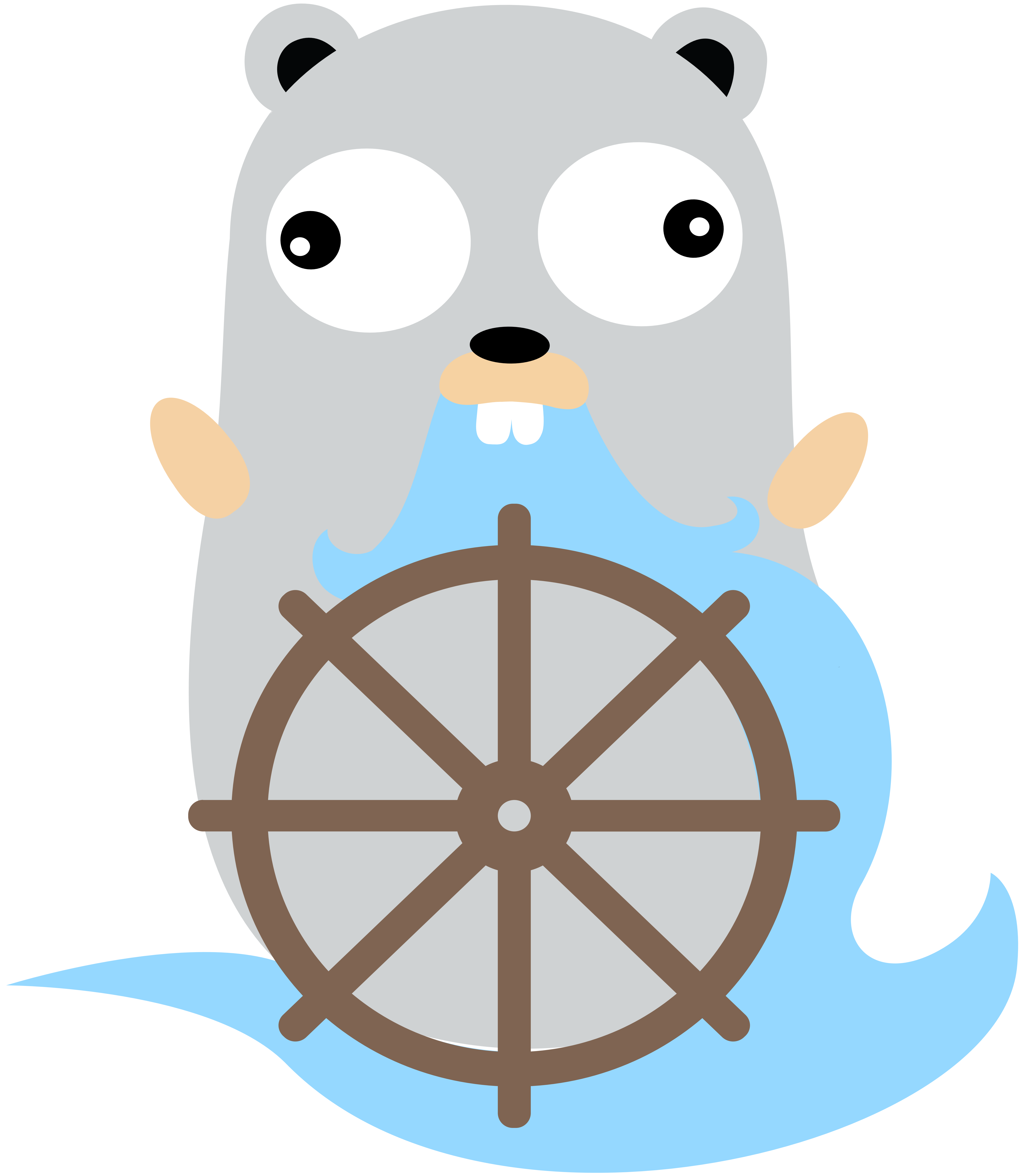
NATS

BoltDB

redis

MySQL™

AND MOST IMPORTANTLY...



IT WASN'T AN OVERNIGHT SUCCESS

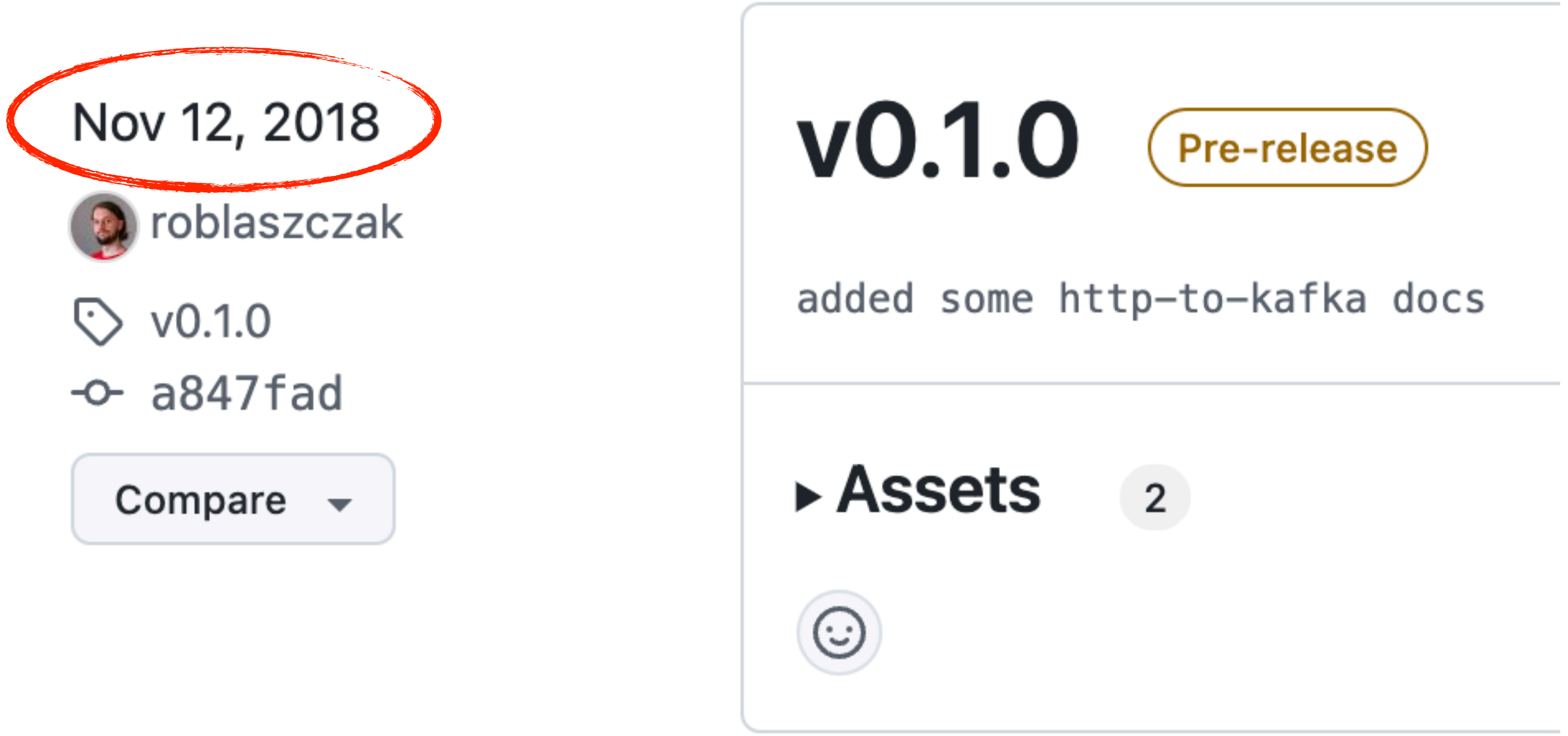
- first commit: 6.5 years ago

IT WASN'T AN OVERNIGHT SUCCESS

- first commit: 6.5 years ago
- v0.1.0 - 6 months after the first commit

IT WASN'T AN OVERNIGHT SUCCESS

- first commit: 6.5 years ago
- v0.1.0 - 6 months after the first commit



Nov 12, 2018
roblaszczak
v0.1.0
a847fad
Compare

v0.1.0 Pre-release

added some http-to-kafka docs

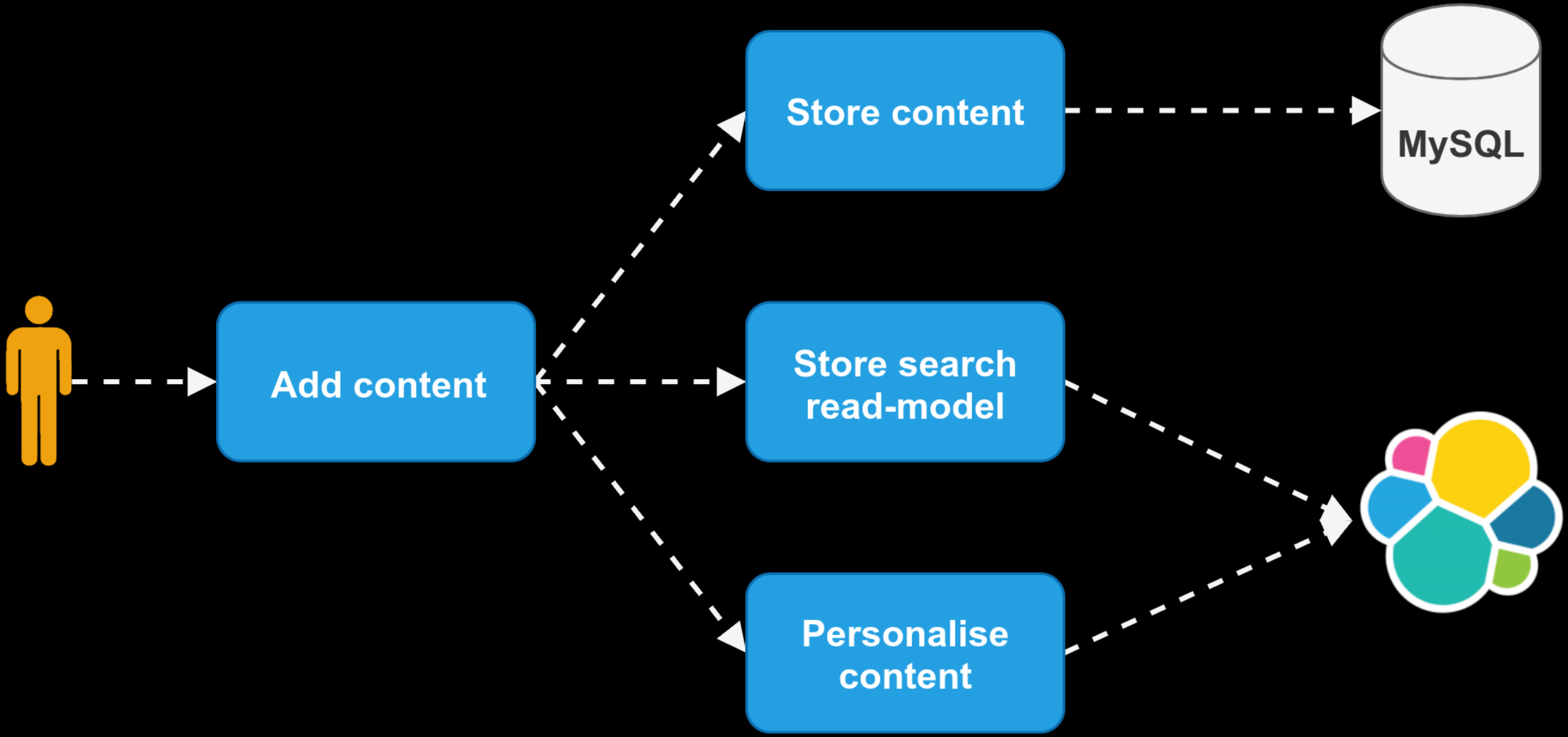
Assets 2

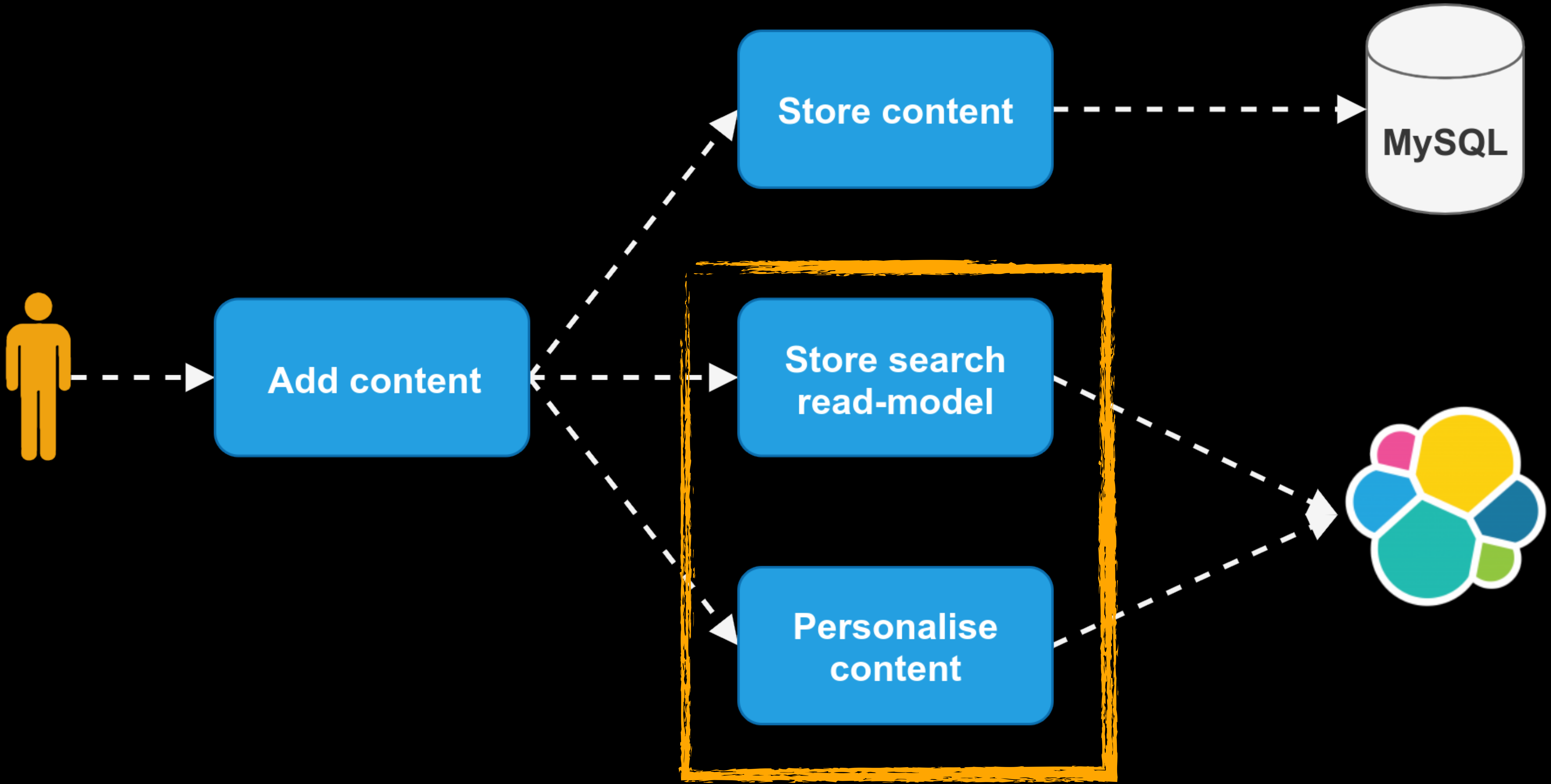
IT WASN'T AN OVERNIGHT SUCCESS

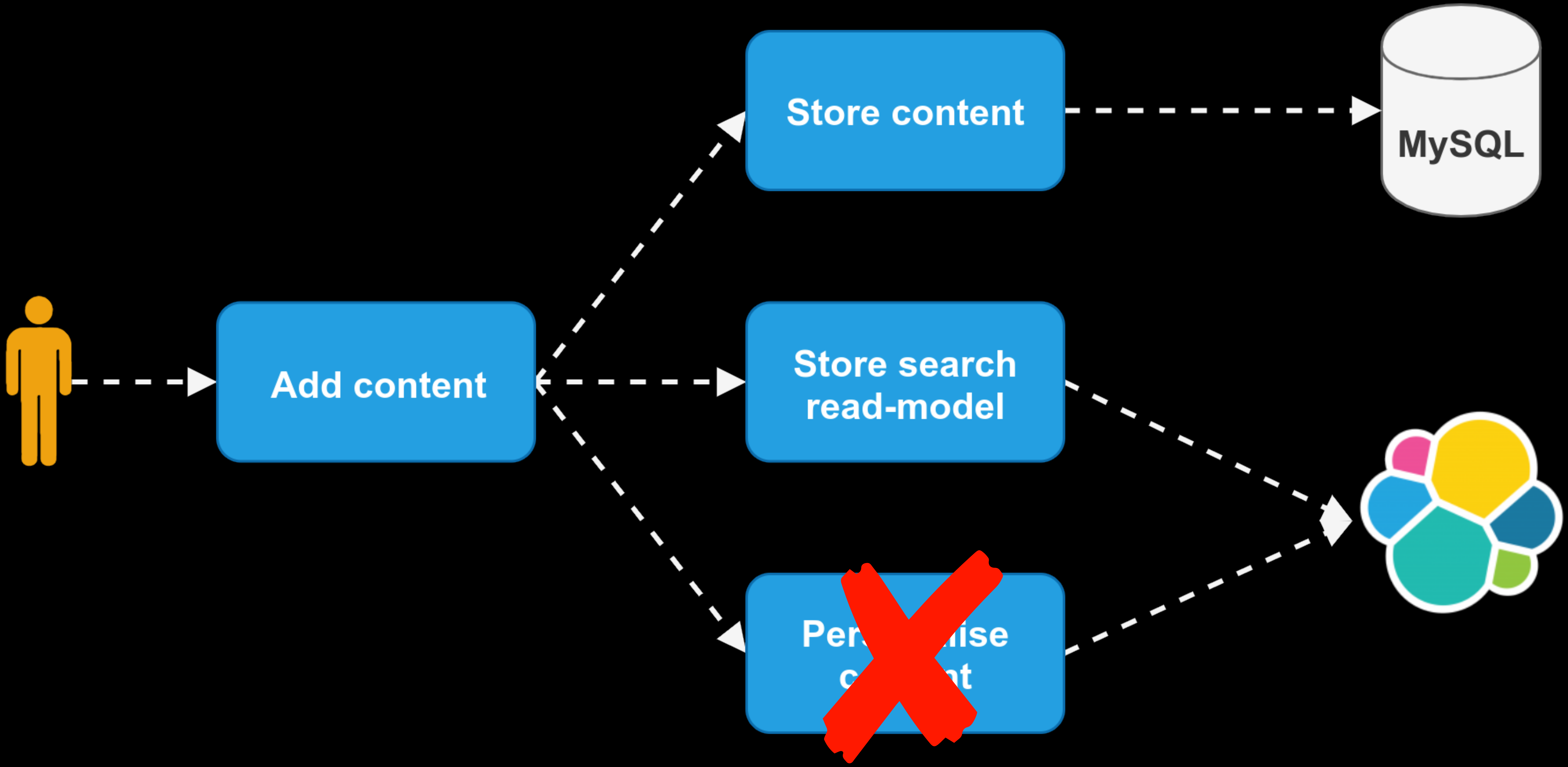
- first commit: 6.5 years ago
- v0.1.0 - 6 months after the first commit
- v1.0 - 500 days after the first commit

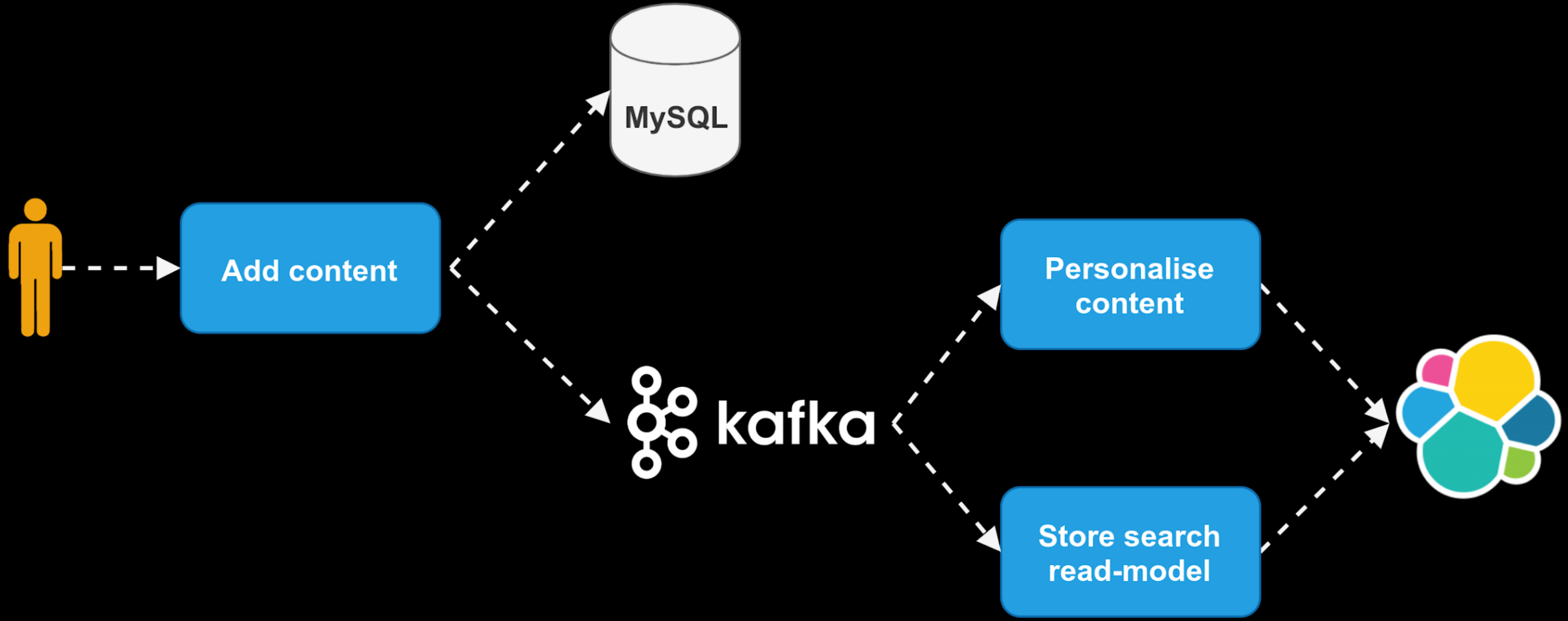
WHAT MADE WATERMILL SUCCESSFUL?

WHAT PROBLEM WATERMILL SOLVES?


Add content
Store content
Store search read-model
Personalise content
MySQL


Add content
Store content
MySQL
Store search read-model
Personalise content


Add content
Store content
MySQL
Store search read-model
Per ise c nt


MySQL
Add content
kafka
Personalise content
Store search read-model

PROBLEMS?

CONSUMER GROUPS

PARTITIONING

MESSAGE ORDERING

AT-LAST-ONCE DELIVERY

MESSAGE ACK AND NACK

POISON QUEUE

NOT LOSING ANY MESSAGE

CONNECTION POOLING

HANDLING TLS

HTTP PROTOCOL

TCP PROTOCOL

DNS RESOLUTION

DEALING WITH PARTIAL READS

HANDLING NETWORK FAILURES

AND IT'S HOW...

added proof of proof of concept

roblaszczak committed on Apr 24, 2018

master · v1.4.0-rc.1 ··· v0.1.0

0 parents commit ddb2d53

32 files changed +1207 -0 lines changed

README.md +5

```
@@ -0,0 +1,5 @@
1 + # GooDDD
2 +
```

FOCUS ON THE PROBLEM

GOALS OF GODDD

- MAKE BUILDING OF MESSAGE/EVENT-BASED SERVICES AS SIMPLE (OR EVEN SIMPLER) THAN HTTP API

GOALS OF GODDD

- MAKE BUILDING OF MESSAGE/EVENT-BASED SERVICES AS SIMPLE (OR EVEN SIMPLER) THAN HTTP API
- EASY TO ADD PUB/SUB IMPLEMENTATION

GOALS OF GODDD

- MAKE BUILDING OF MESSAGE/EVENT-BASED SERVICES AS SIMPLE (OR EVEN SIMPLER) THAN HTTP API
- EASY TO ADD PUB/SUB IMPLEMENTATION
- IT SHOULD BE AN EXTENSIBLE LIBRARY

UNIX PHILOSOPHY

- WRITE PROGRAMS **THAT DO ONE THING AND DO IT WELL.**
- WRITE PROGRAMS **TO WORK TOGETHER.**
- WRITE PROGRAMS TO HANDLE ~~**TEXT STREAMS**~~ **MESSAGES,**

  BECAUSE THAT IS A UNIVERSAL INTERFACE.

UNIX PHILOSOPHY (1978)

- WRITE PROGRAMS **THAT DO ONE THING AND DO IT WELL.**
- WRITE PROGRAMS **TO WORK TOGETHER.**
- WRITE PROGRAMS TO HANDLE ~~**TEXT STREAMS**~~ **MESSAGES,**

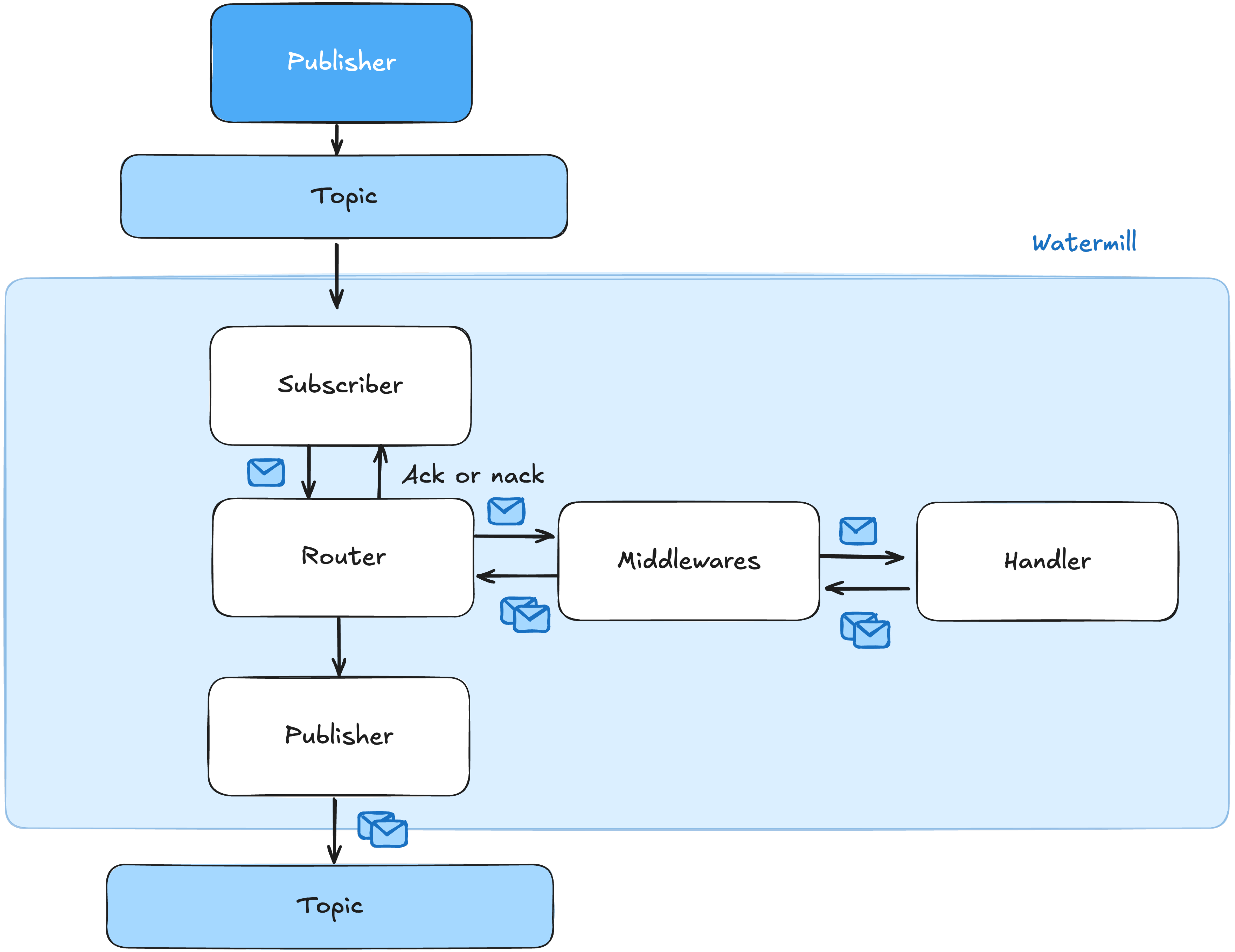
  BECAUSE THAT IS A UNIVERSAL INTERFACE.

```go
type Message struct {
  UUID string

  Metadata map[string]string

  Payload []byte
}
```

```go
type Publisher interface {
    Publish(topic string, messages ...*Message) error
    Close() error
}
```

```go
type Publisher interface {
    Publish(topic string, messages ...*Message) error
    Close() error
}
type Subscriber interface {
  Subscribe(
     ctx context.Context,
     topic string,
  ) (<-chan *Message, error)
    Close() error
}
```

```go
type HandlerFunc func(msg *Message) ([]*Message, error)
```

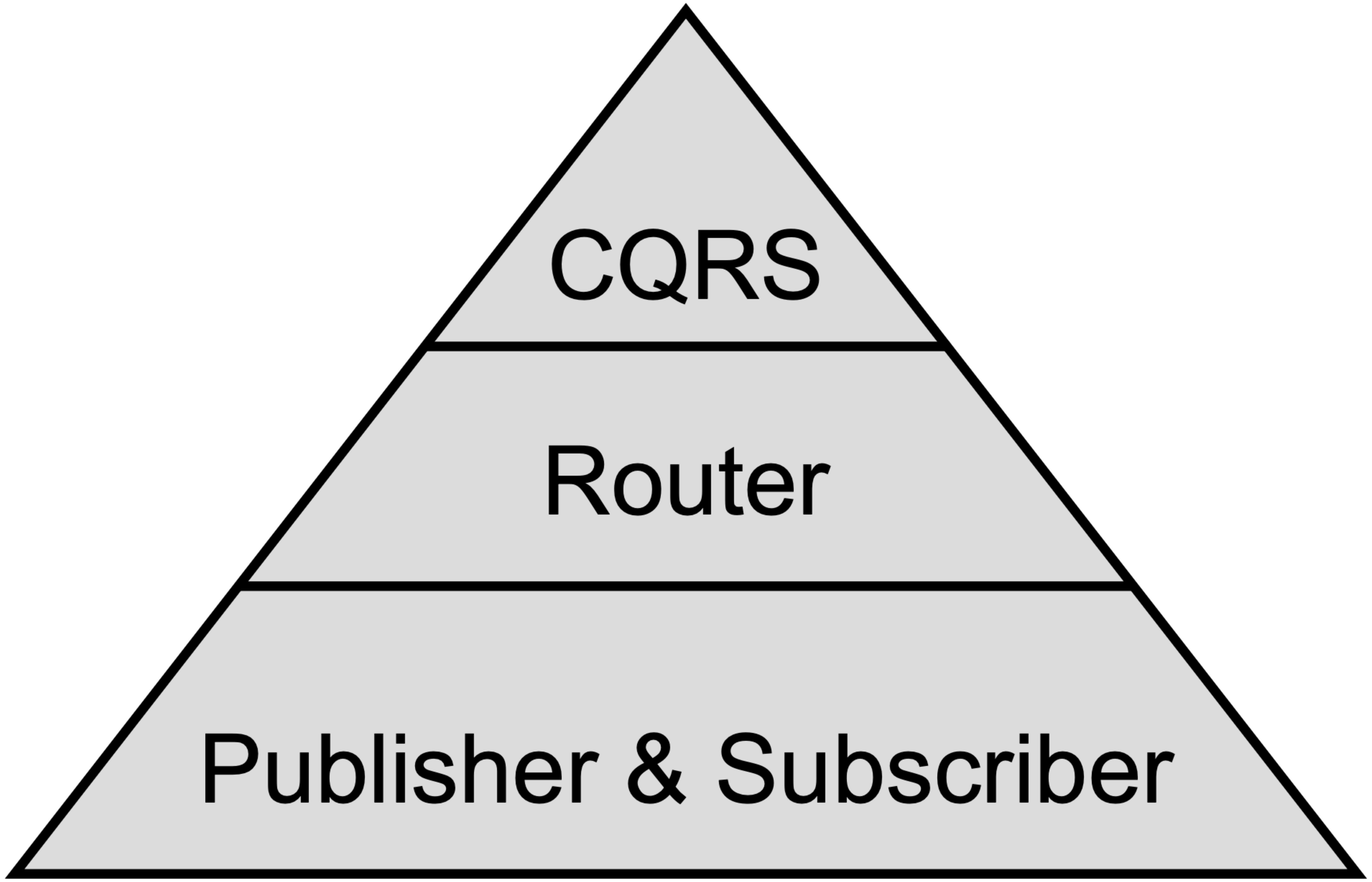

Publisher
Topic
Watermill
Subscriber
Ack or nack
Router
Middlewares
Handler
Publisher
Topic

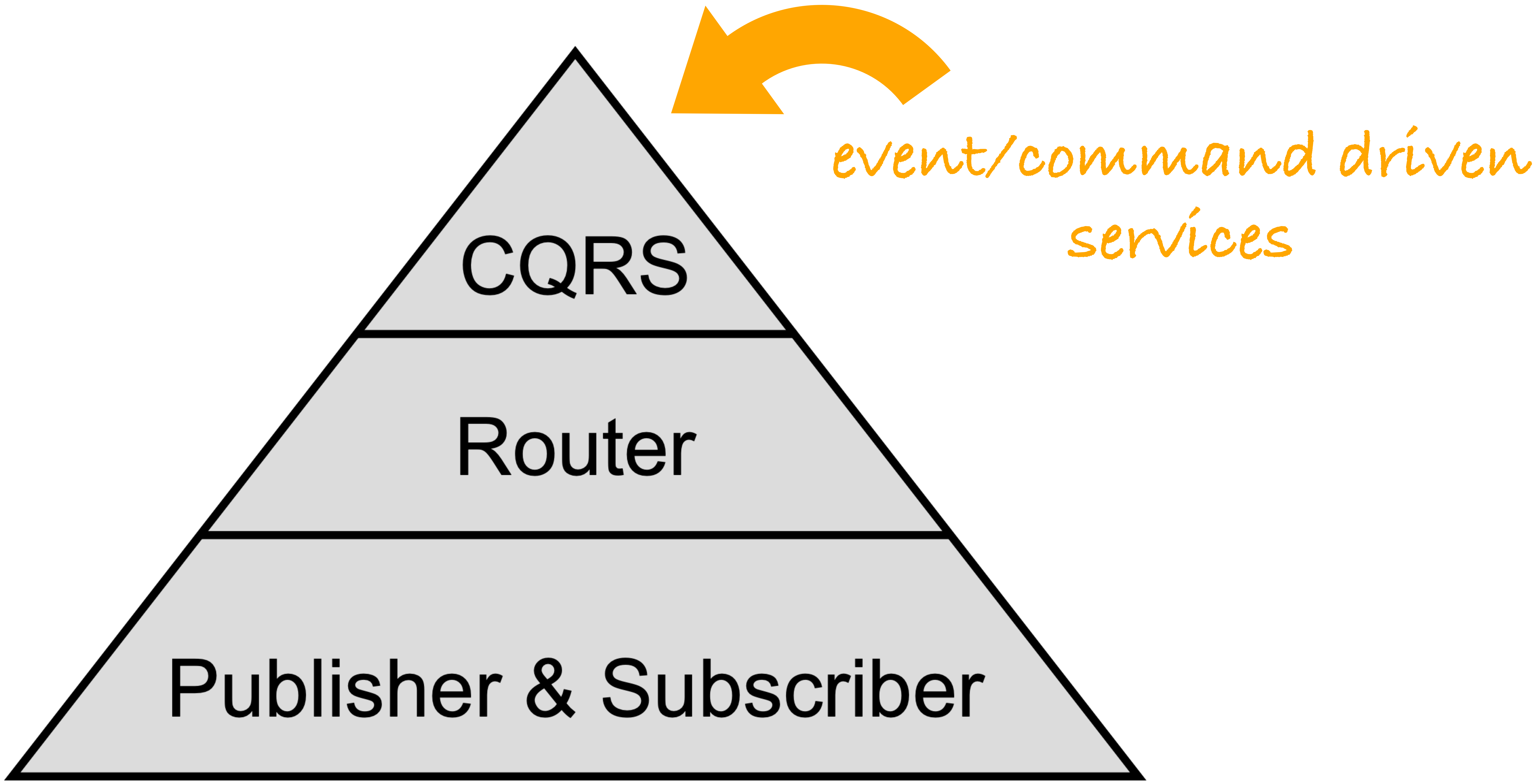
MIDDLEWARES

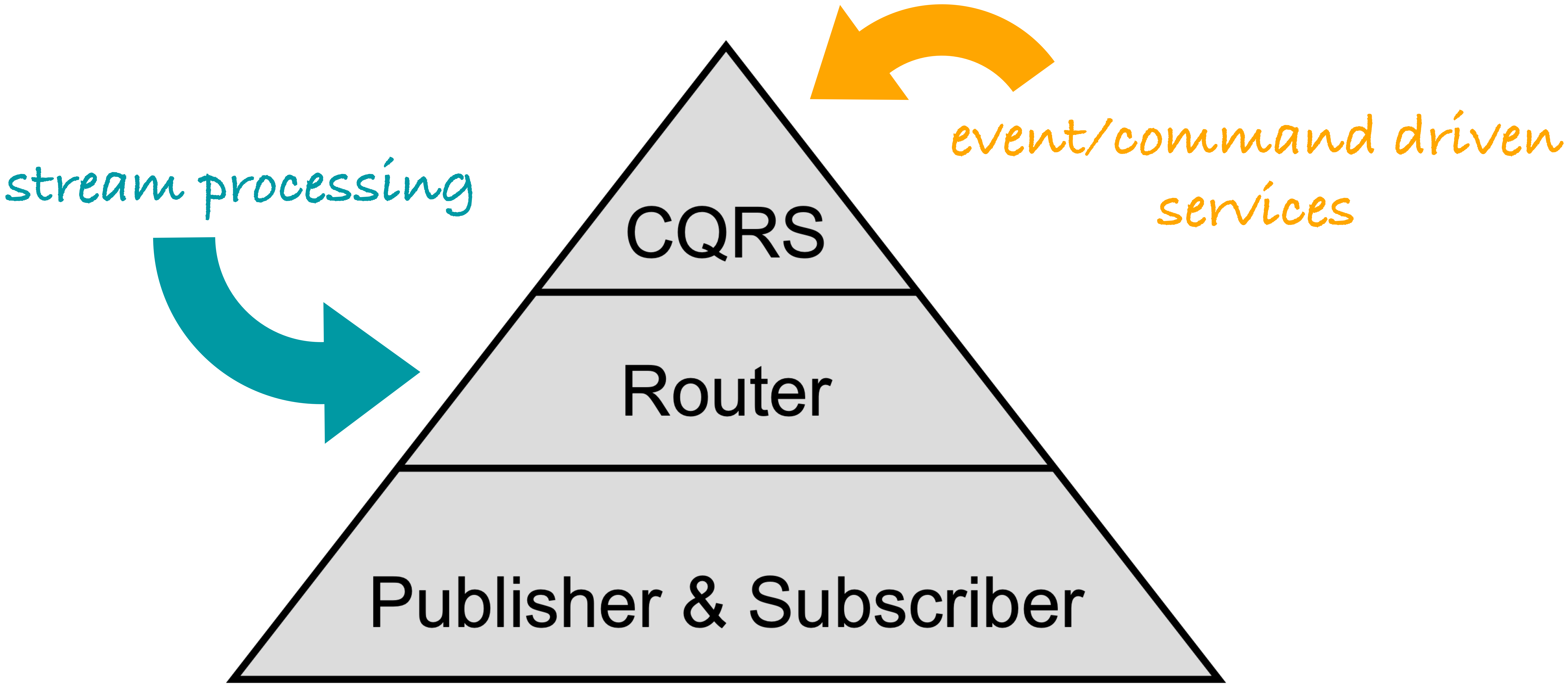
- TIMEOUT
- CORRELATION
- LOGGING
- POISON QUEUE
- RETRIES
- THROTTLING
- CIRCUIT BREAKER
- DUPLICATOR
- RANDOM FAIL
- WRITE YOUR OWN!

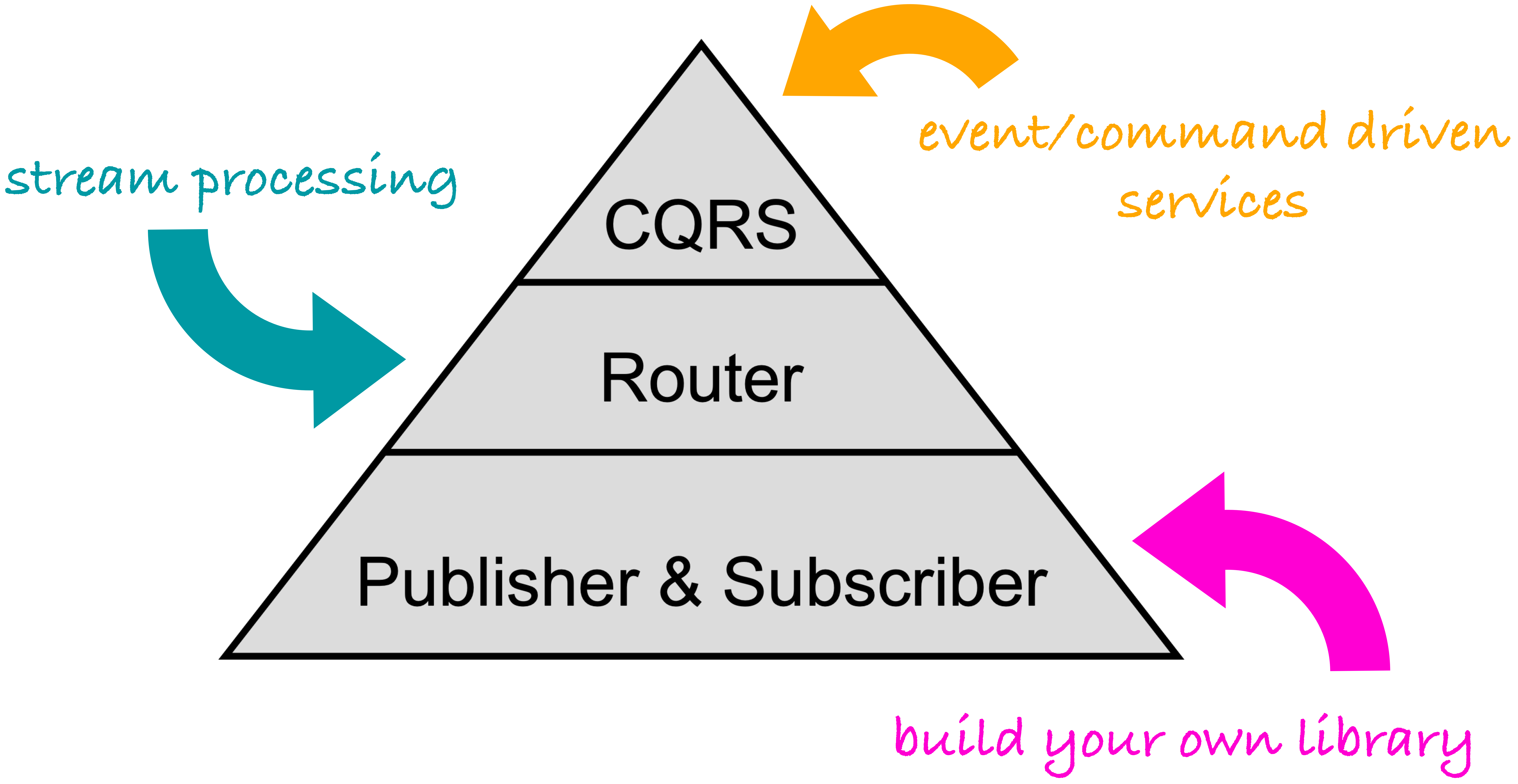
```go
type Message struct {
  UUID string

  Metadata map[string]string

  Payload []byte
}
```

```
type OrderPlaced struct {
  OrderID uuid.UUID `json:"order_id"
}


cqrs.NewEventHandler(
  "send_order",
  func(ctx context.Context, op *OrderPlaced) error {
    return orders.SendOrder(op.OrderID)
   },
)
```


CQRS
Router
Publisher & Subscriber


CQRS
Router
Publisher & Subscriber
event/command driven services


stream processing
event/command driven services
CQRS
Router
Publisher & Subscriber


stream processing
event/command driven services
CQRS
Router
Publisher & Subscriber
build your own library

GOING TOWARDS v1.0

TESTING STRATEGY

TESTING STRATEGY

- MAKE LONG-TERM DEVELOPMENT EASIER

TESTING STRATEGY

- MAKE LONG-TERM DEVELOPMENT EASIER
- ADDING NEW PUB/SUB EASILY

TESTING STRATEGY

- MAKE LONG-TERM DEVELOPMENT EASIER
- ADDING NEW PUB/SUB EASILY
- DO NOT RE-IMPLEMENT ALL EDGE CASES

```go
type Publisher interface {
    Publish(topic string, messages ...*Message) error
    Close() error
}
type Subscriber interface {
  Subscribe(
     ctx context.Context,
     topic string,
  ) (<-chan *Message, error)
    Close() error
}
```

```go
func TestPublishSubscribe(t *testing.T) {
    features := tests.Features{
        ConsumerGroups:      true,
        ExactlyOnceDelivery: false,
        GuaranteedOrder:     false,
        Persistent:          true,
    }

    tests.TestPubSub(
        t,
        features,
        createPubSub,
        createPubSubWithConsumerGrup,
    )
}
```

ESTABLISH PUBLIC INTERFACES

ESTABLISH PUBLIC INTERFACES

- **NO V2 PLANNED- NO BREAKING CHANGES WITHIN MAJOR**

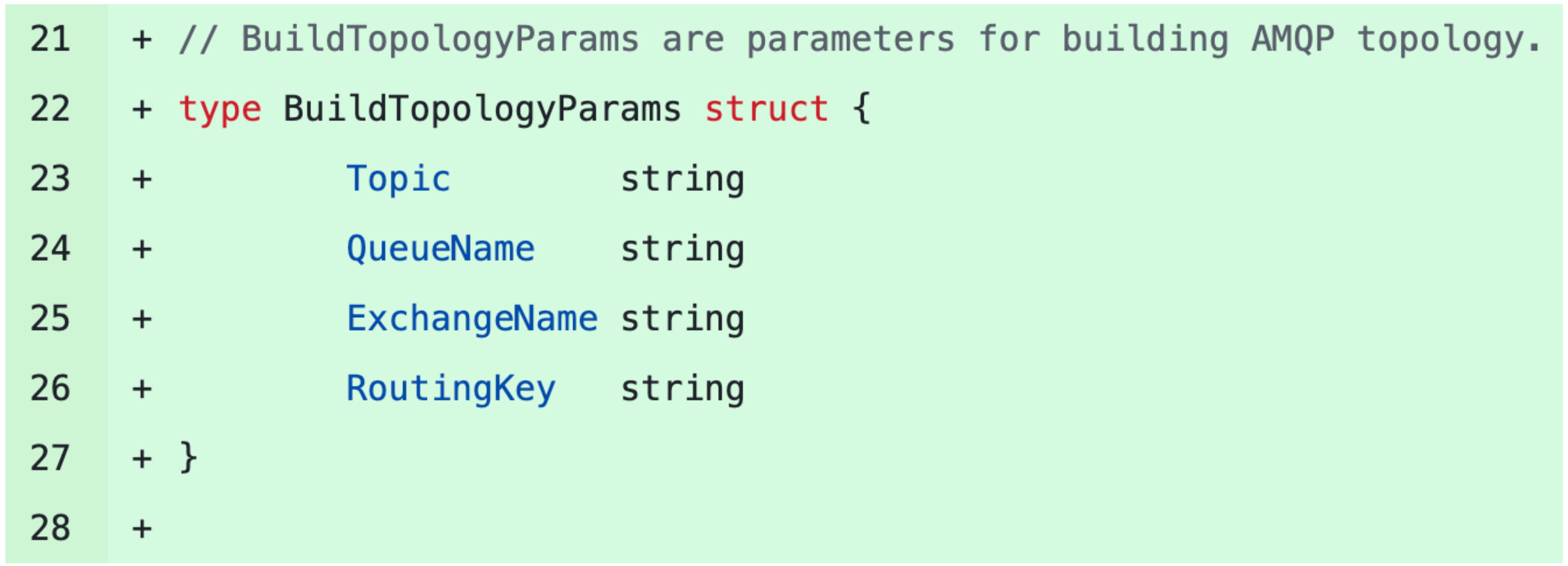
ESTABLISH PUBLIC INTERFACES

- NO V2 PLANNED- NO BREAKING CHANGES WITHIN MAJOR
- ASK OTHERS FOR FEEDBACK AND DOGFOOD

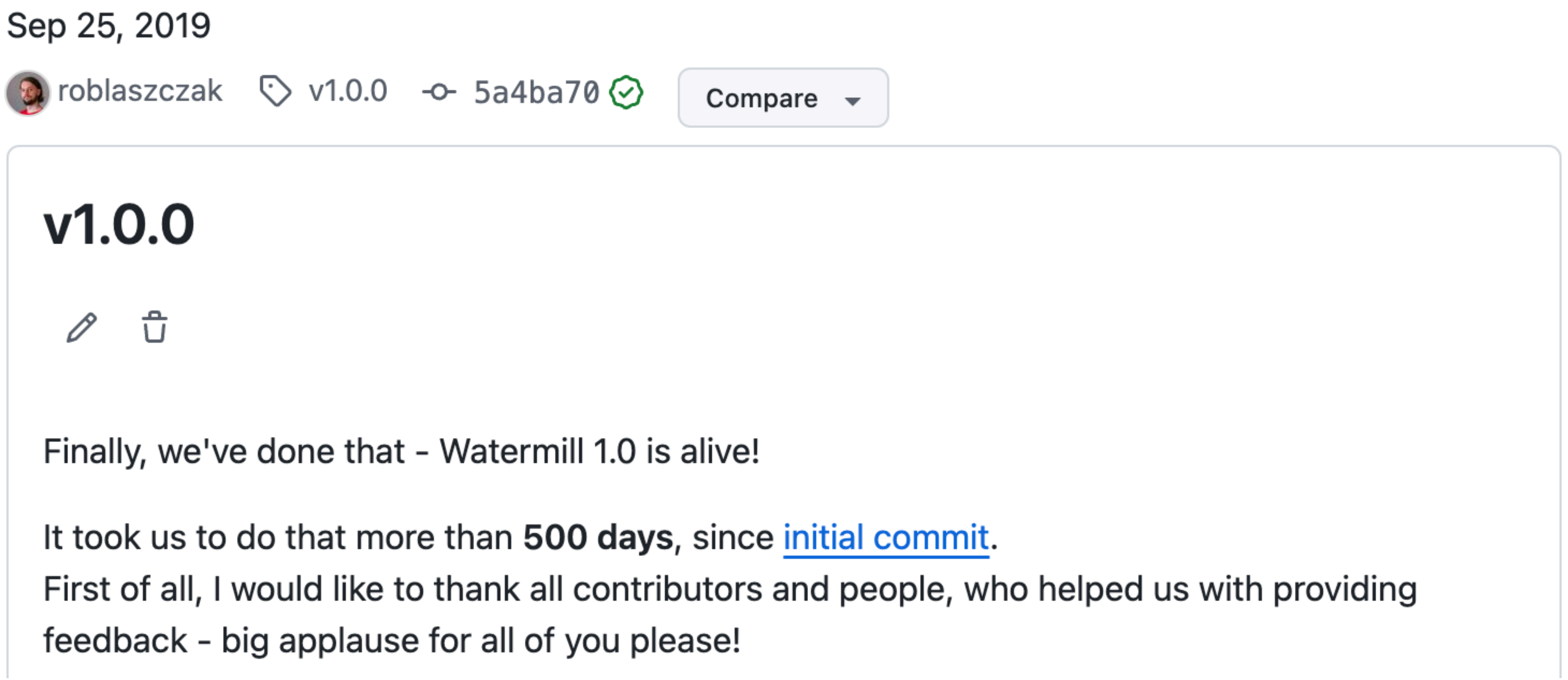
LESSONS LEARNED?

```diff
 type TopologyBuilder interface {
-	BuildTopology(channel *amqp.Channel, queueName string, exchangeName string, config Config, logger watermill.LoggerAdapter) error
 	ExchangeDeclare(channel *amqp.Channel, exchangeName string, config Config) error
 }
```

```diff
 type TopologyBuilder interface {
+	BuildTopology(channel *amqp.Channel, params BuildTopologyParams, config Config, logger watermill.LoggerAdapter) error
 	ExchangeDeclare(channel *amqp.Channel, exchangeName string, config Config) error
 }
```



```diff
+ // BuildTopologyParams are parameters for building AMQP topology.
+ type BuildTopologyParams struct {
+ 	Topic        string
+ 	QueueName    string
+ 	ExchangeName string
+ 	RoutingKey   string
+ }
+
```



Sep 25, 2019

roblaszczak v1.0.0 5a4ba70 Compare

v1.0.0

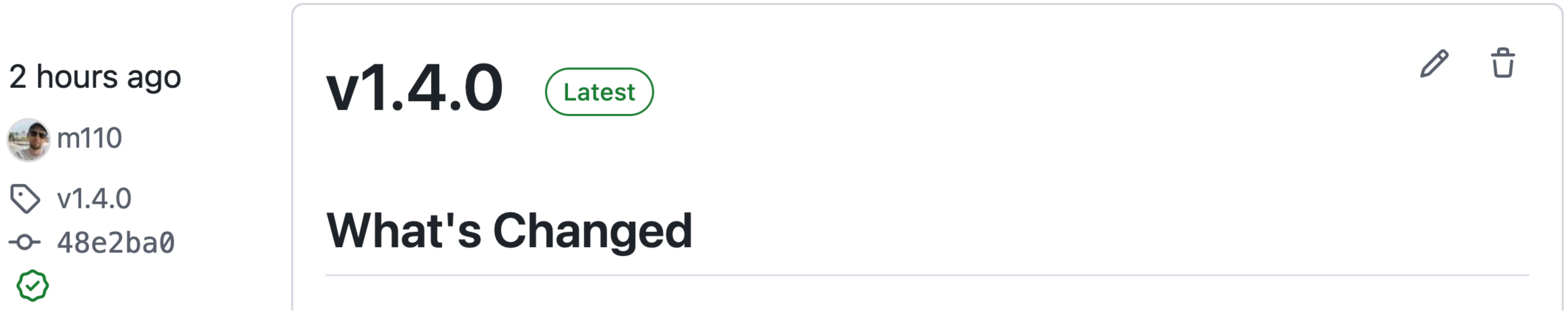
Finally, we've done that - Watermill 1.0 is alive!

It took us to do that more than **500 days**, since initial commit.
First of all, I would like to thank all contributors and people, who helped us with providing feedback - big applause for all of you please!

#1 RULE OF LIVE CODING

#1 RULE OF LIVE CODING

DON’T DO LIVE CODING



2 hours ago

m110

v1.4.0

48e2ba0

v1.4.0 Latest

What's Changed

SOME FEATURES SINCE V1.0

- REQUEST/REPLY SUPPORT
- FORWARDER COMPONENT (OUTBOX PATTERN)
- MORE PUB/SUBS (AWS, POSTGRES QUEUE)
- OUT-OF-THE-BOX SSE SUPPORT
- POISON QUEUE CLI
- DOCS…

Watermill

Building event-driven applications the easy way in Go.

Get Started See on GitHub

Easy to Use

Watermill is easy to grasp, even for junior developers.

Universal

Build event-driven architecture, CQRS, Event Sourcing or just stream Postgres to Kafka.

Fast

Watermill was designed to process hundreds of thousands of messages per second.

THANKS FOR ALL CONTRIBUTORS

THANKS!

https://tdl.is/golab24/