RabbitMQ

How to make microservices talk

Ms.C. **João Daher**

Computer Science @ UFLA
Masters Artificial Intelligence @ UNIFEI
Backend Developer @ eduK

What?

Message broker

Written in Erlang

Implements AMQP Protocol

Why?

Deliver later

Asynchronous processing

Loose coupling

Load balancing

Open source

Alternatives?

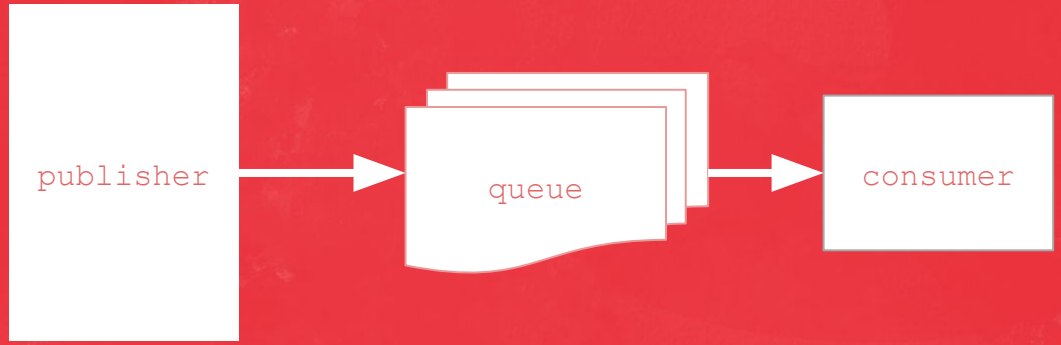
Redis

AWS **SNS**

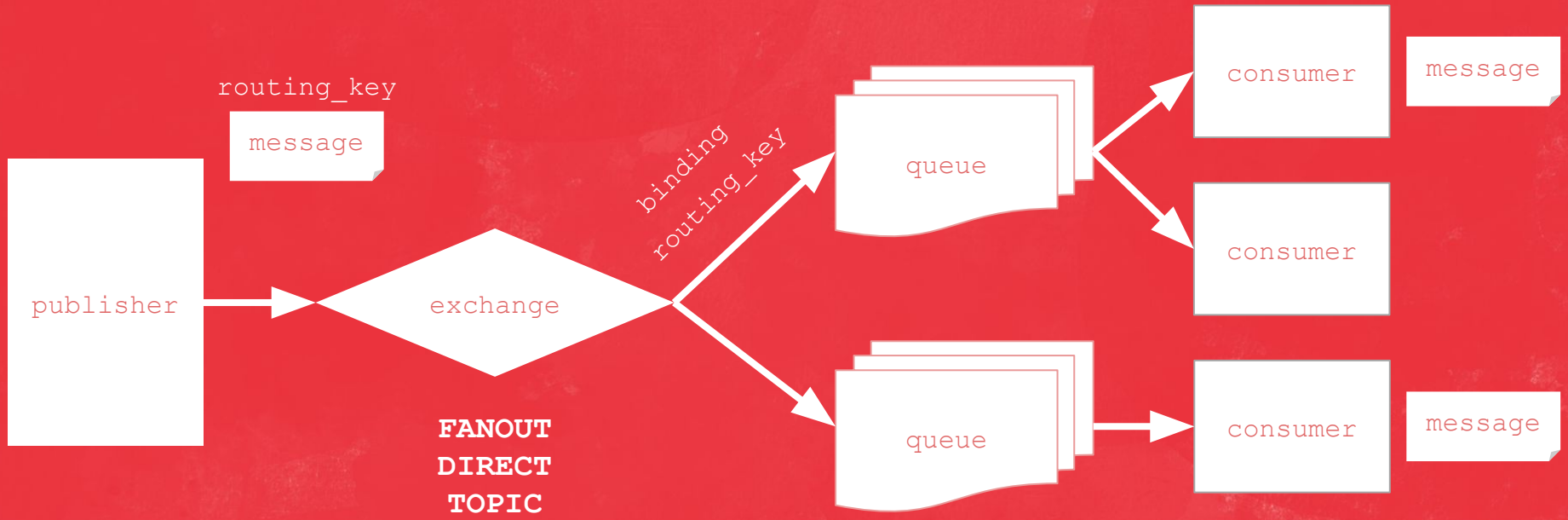
Apache **Kafka**

Google **Pubsub**

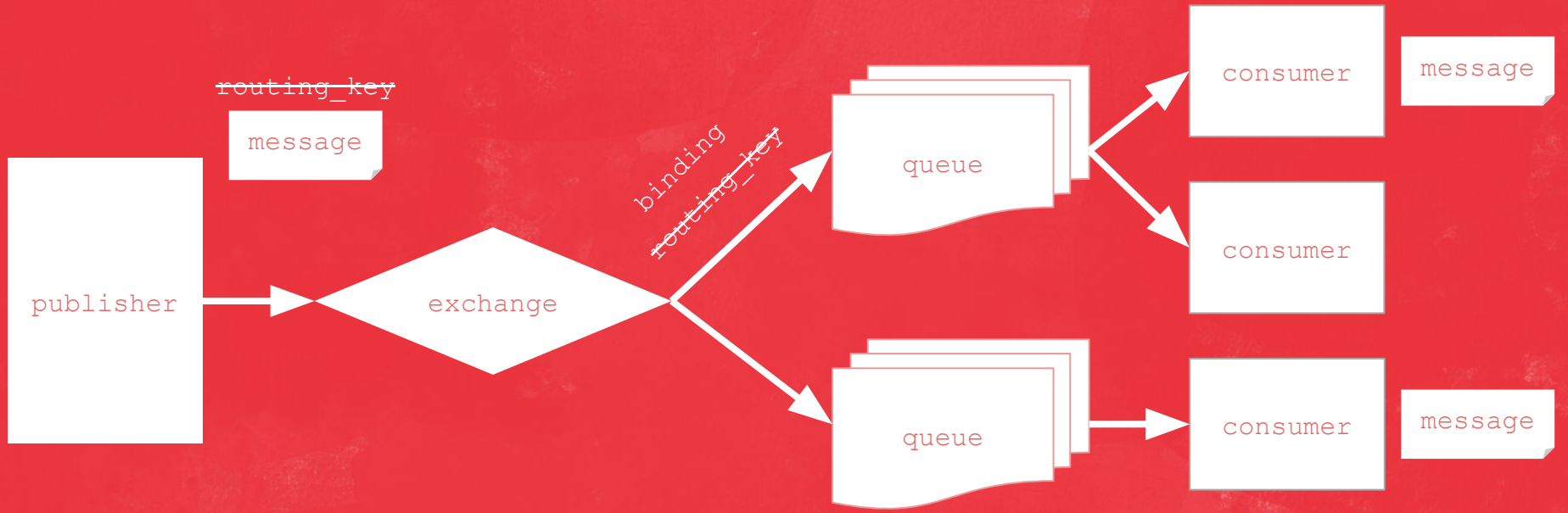
PubSub?



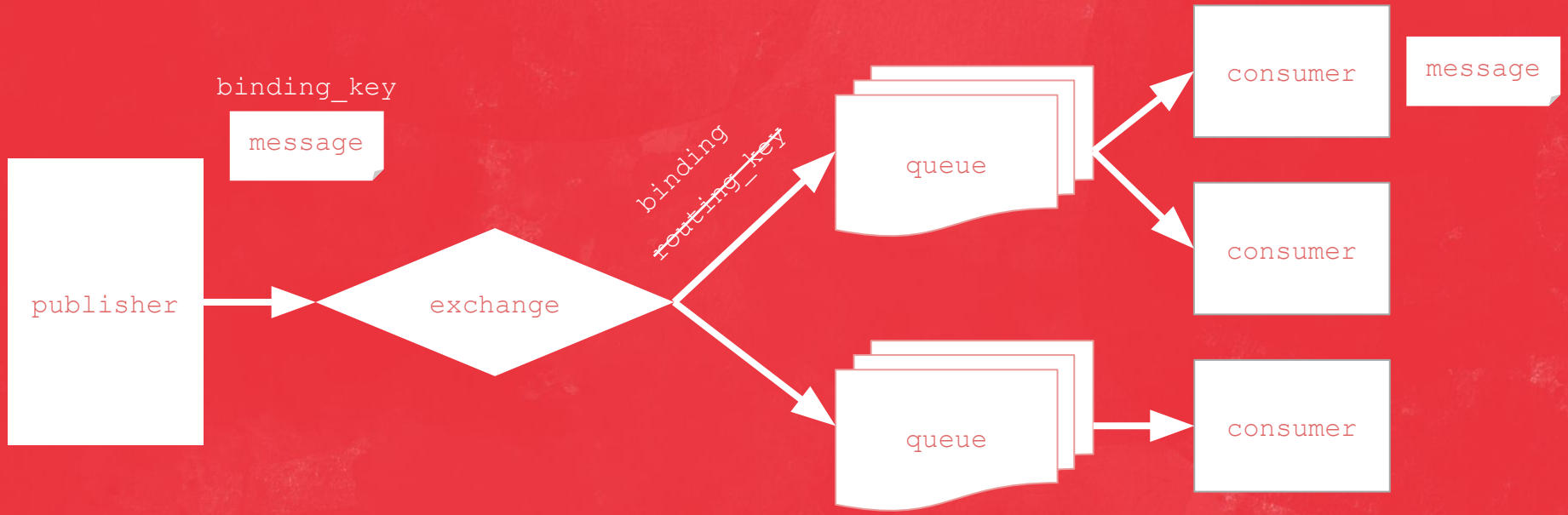
AMQP?


routing_key
message
publisher
exchange
binding
routing_key
queue
queue
consumer
consumer
consumer
message
message
FANOUT
DIRECT
TOPIC

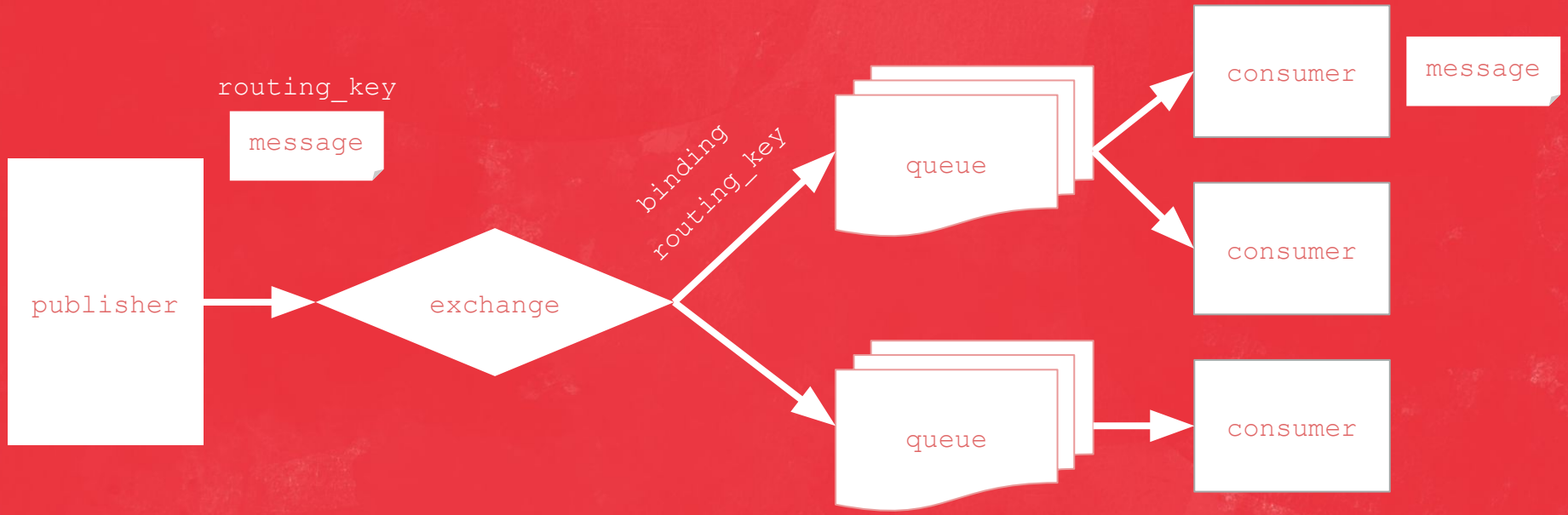
Fanout?


~~routing_key~~
message
publisher
exchange
binding
~~routing_key~~
queue
queue
consumer
consumer
consumer
message
message

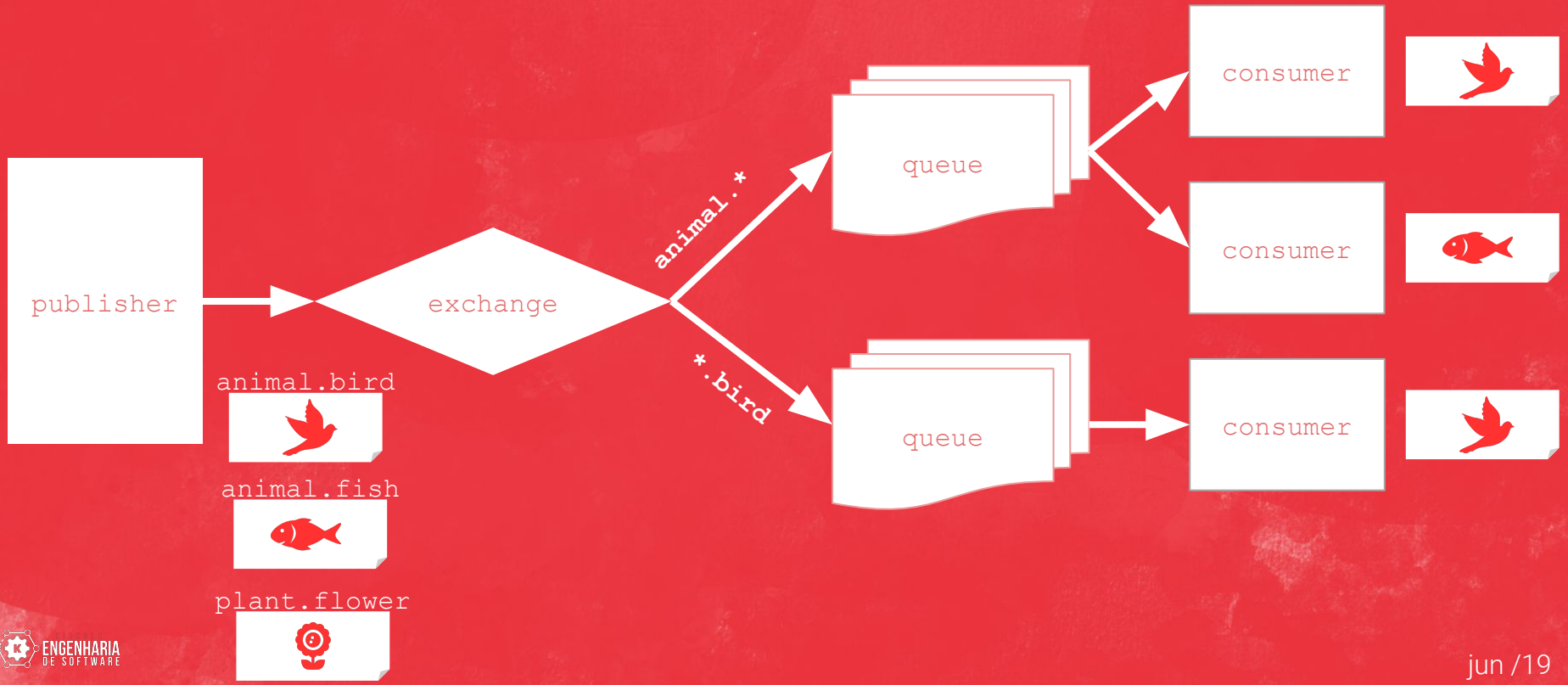
Direct?


binding_key
message
publisher
exchange
binding
~~routing_key~~
queue
queue
consumer
consumer
consumer
message

Topic?


routing_key
message
publisher
exchange
binding
routing_key
queue
queue
consumer
consumer
consumer
message

How does it work?



Python?

Pika

`pip install pika`

- Small & Compact
- Low Level

Kombu

`pip install kombu`

- Retrying
- Failover
- Connection Pool
- High Level
- Celery Project

Publishing?

Kombu Producer

```
from kombu import Connection, Exchange, Producer

conn = Connection("amqp://localhost:5672/")
my_exchange = Exchange(name='nature', type='topic')
producer = Producer(
    exchange=my_exchange,
    routing_key="animal.bird",
    channel=conn.channel(),
)
producer.publish(message={'bird_name': dove})
```

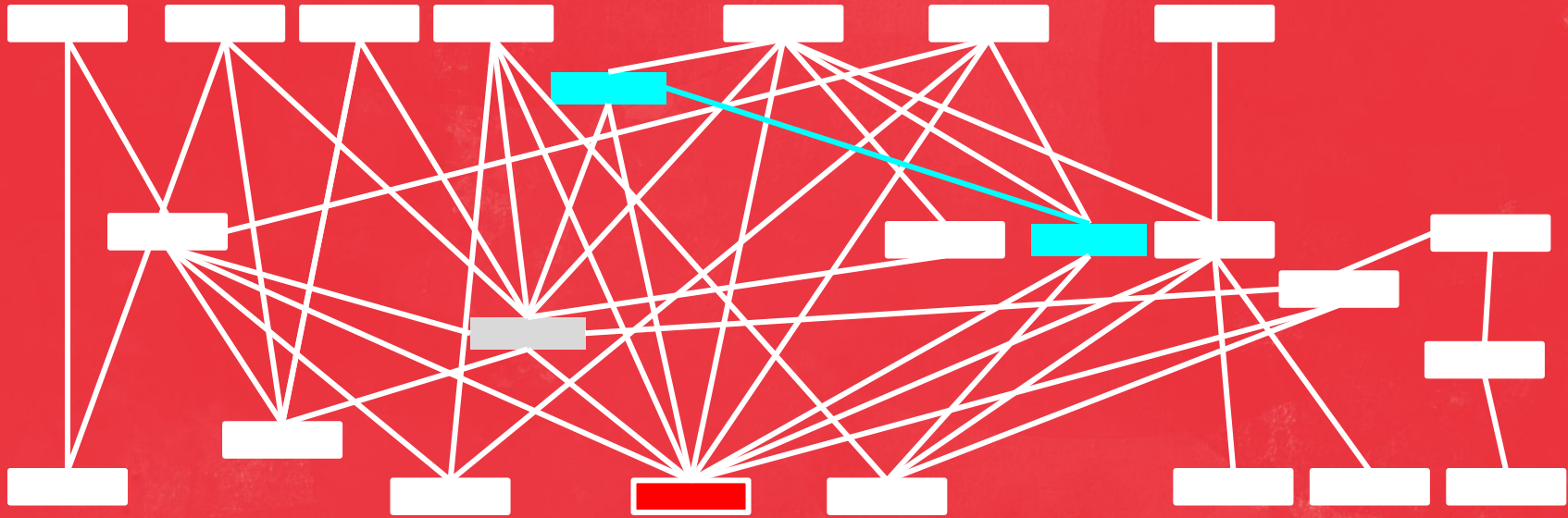
animal.bird

Consuming?

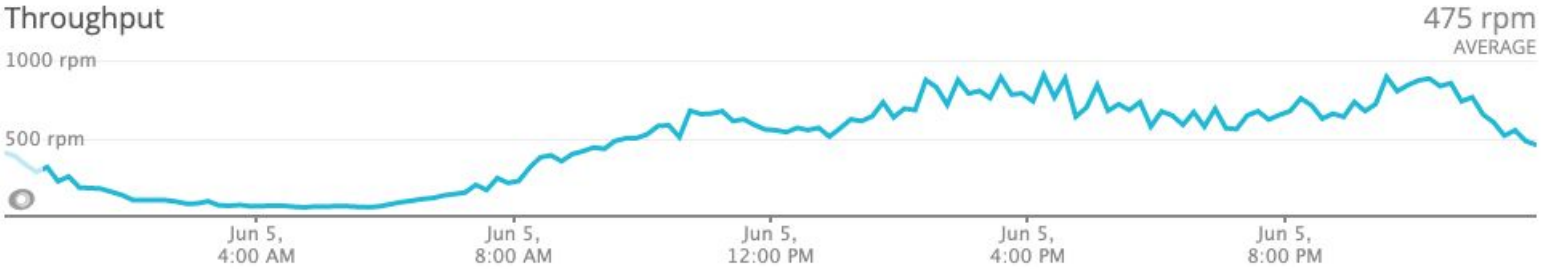
Kombu Consumer

```
from kombu import Connection, Exchange, Queue, Consumer

conn = Connection("amqp://localhost:5672/")
my_exchange = Exchange(name='nature', type='topic')
bird_queue = Queue(name="birds_only", exchange=exchange,  routing_key="*.bird")
bird_queue.maybe_bind(conn)
bird_queue.declare()
with Consumer(conn, queues=bird_queue, callbacks=[process_message]):
    conn.drain_events()

def process_message(body, message):
    print(f"Do some magic with this: {body}")
    message.ack()
```

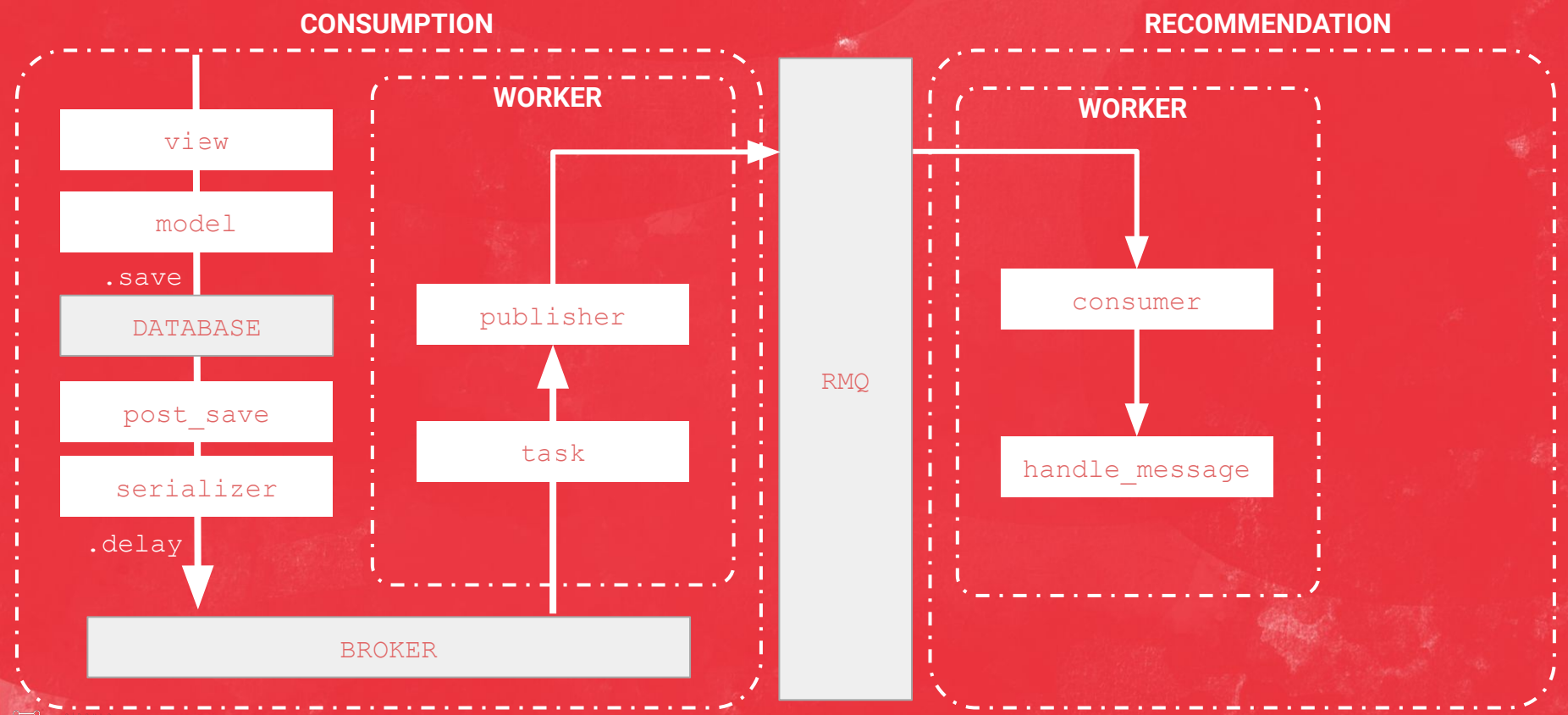
The Problem IRL?



The Problem IRL?



Solution?


CONSUMPTION
RECOMMENDATION
WORKER
WORKER
view
model
.save
DATABASE
post_save
serializer
.delay
publisher
task
RMQ
consumer
handle_message
BROKER

Pattern?

```
class Watch(Model):
    user_id = IntegerField()
    lesson_id = IntegerField()
    position = IntegerField()


class WatchSerializer(ModelSerializer):
    Class Meta:
        model = Watch

@receiver(post_save, sender=Watch)
def publish_watch(sender, instance, created, **kwargs):
    PublishModelTask().delay(
        data=<body>,
        routing_key=<routing_key>,
        exchange=<exchange>
     )
```

Message

body

```
{
    user_id: 42,
    lesson_id: 100,
    position: 314
}
```

routing_key

```
created
```

exchange

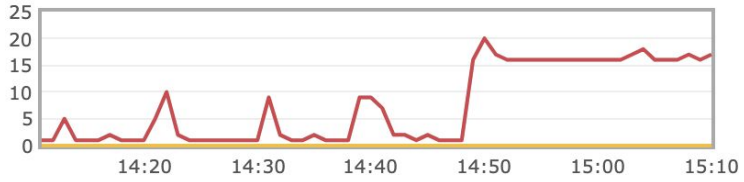
```
watch
```

queue

```
recommendation.watch.*
```

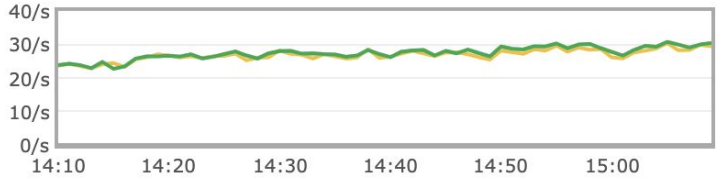
Results?

Queued messages last hour ?



Ready	0
Unacked	16
Total	16

Message rates last hour ?



Publish	29/s
Deliver (manual ack)	30/s
Deliver (auto ack)	0.00/s

Consumer ack	30/s
Redelivered	0.00/s
Get (manual ack)	0.00/s
Get (auto ack)	0.00/s

edu K

CÍRCULO
ENGENHARIA
DE SOFTWARE

VIVA
DA SUA
PAIXÃO

CONTACT
joao@daher.dev

WE ARE HIRING
http://**eduk.breezy.hr**