



Solution Architect





Somkiat
Home

Somkiat Puisungnoen

Update Info 1

View Activity Log 10+

...

Timeline

About

Friends 3,138

Photos

More ▾

When did you work at Opendream?

×

...

22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชำนานุกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post

Photo/Video

Live Video

Life Event

What's on your mind?

Public ▾

Post

Somkiat Puisungnoen

15 mins · Bangkok · 🌐 ▾

Java and Bigdata

...



Facebook interface for the page **somkiat.cc**. The top navigation bar includes the Facebook logo, the page name, a search bar, and icons for Home, Messages, Notifications (3), Insights, Publishing Tools, Settings, and Help.

The main content area features a large video player showing a man in a white Superman t-shirt with "SOMKIAT.CC" printed on it, posing against a white background. Below the video are buttons for "Liked", "Following", "Share", and a menu icon. A blue call-to-action button reads "Help people take action on this Page." with a close icon. Another blue button below it says "+ Add a Button".

The left sidebar contains the page name **somkiat.cc**, the handle **@somkiat.cc**, and a menu with options: Home, Posts, Videos, and Photos.



**[https://github.com/up1/
course-rabbitmq-2025](https://github.com/up1/course-rabbitmq-2025)**





<https://www.rabbitmq.com/>



Topics

Messaging queue
Introduction to RabbitMQ
Use cases
Architecture of RabbitMQ
RabbitMQ patterns
Clustering
Monitoring and observability
Development



Messaging Queue



Best Solution ?



Introduction to RabbitMQ





How RabbitMQ works ?



RabbitMQ works

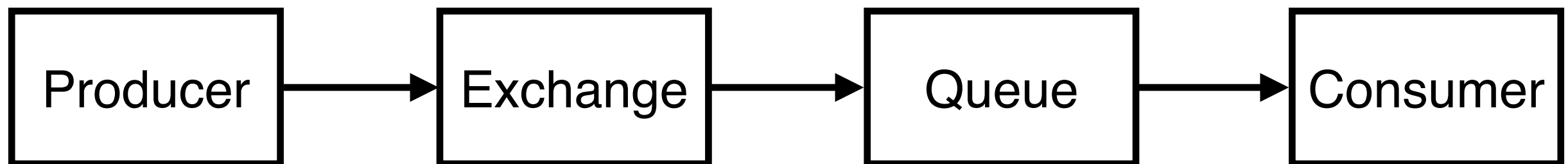
Queue-based messaging model

Messages are published by **producers**

Routed by **exchange**

Delivery to **consumers** through **queues**

Queue is a buffer that stores messages



Queue ?

Storage to store **ordered** collection of messages

Allow enqueue and dequeue

FIFO (First-in First-out)

Stored messages until a consumer is available to process

Durable

Exclusive

Auto-delete



Queue Types ?

Classic (default)

Quorum

Stream



Classic queue

Classic (default)

FIFO (First-in, First-out) message delivery

Suitable for non-critical application

General purpose and background processing

Data loss is not major concern



Quorum queue

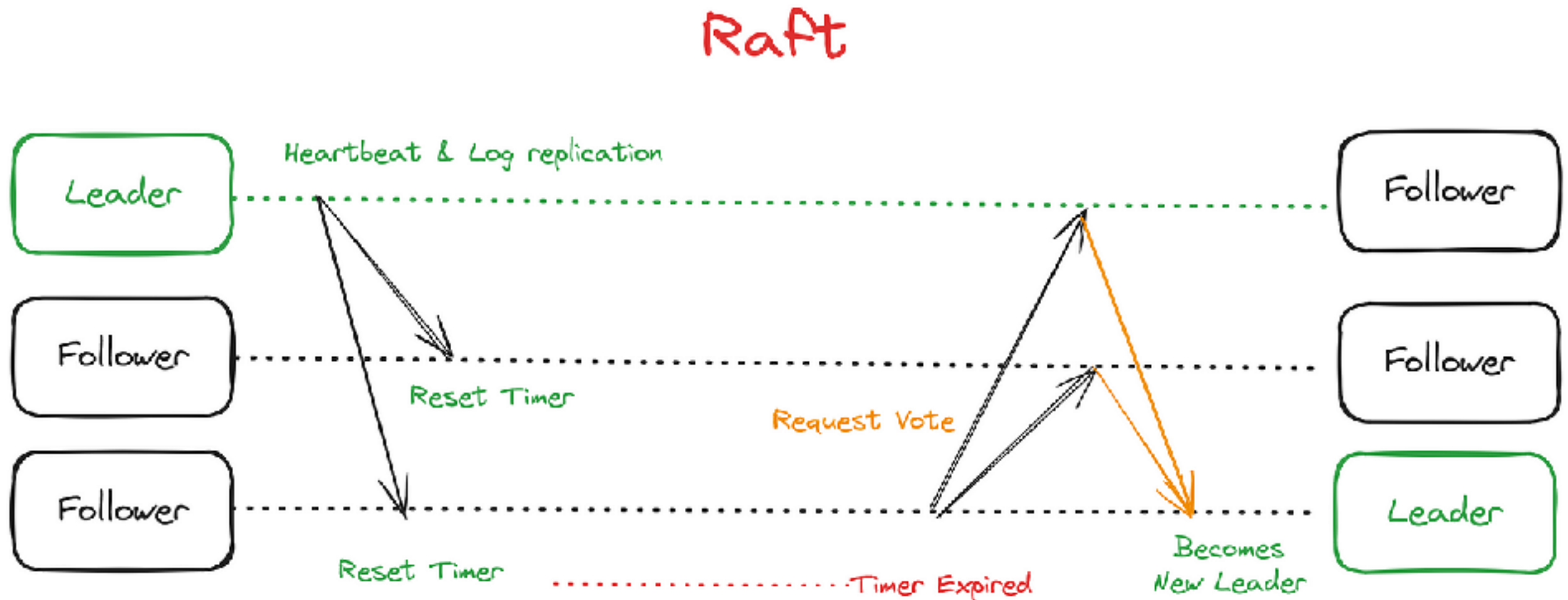
Provide high availability and data safety
Replication using Raft consensus algorithm
Replicated data across multiple nodes

High latency !!

Data loss is unacceptable
(Financial data and sensitive data)



Raft consensus ?



[https://en.wikipedia.org/wiki/Raft_\(algorithm\)](https://en.wikipedia.org/wiki/Raft_(algorithm))



how does RAFT WORK?

RAFT IS A
CONSENSUS ALGORITHM

used in many
projects in the
real world -
ex. etcd

a process
for getting
multiple machines
to "agree"!

WHY? Any part of
a distributed
system can fail!
Consensus across
several nodes gives
us backups of data.

WHAT ABOUT NETWORK
PARTITIONS?!

In a partition,
if the leader
falls in the smaller
side, a new leader is
elected within the majority
side. When the partition
ends, messages received
by the minority side are
discarded, and those nodes
converge their state to
match the majority's.

WHEN A
NEW WRITE
HAPPENS:



Any node can
accept writes!



Message
forwarded
to leader
node

Message is
queued...



COPY
me!



ACK!

...while followers
copy it to their logs



ACK



When a quorum have
copied the message, leader
"commits" the new data
& will now return it on read

Followers also commit
so they return new data!



@denisequ21 with @mt165

[https://en.wikipedia.org/wiki/Raft_\(algorithm\)](https://en.wikipedia.org/wiki/Raft_(algorithm))



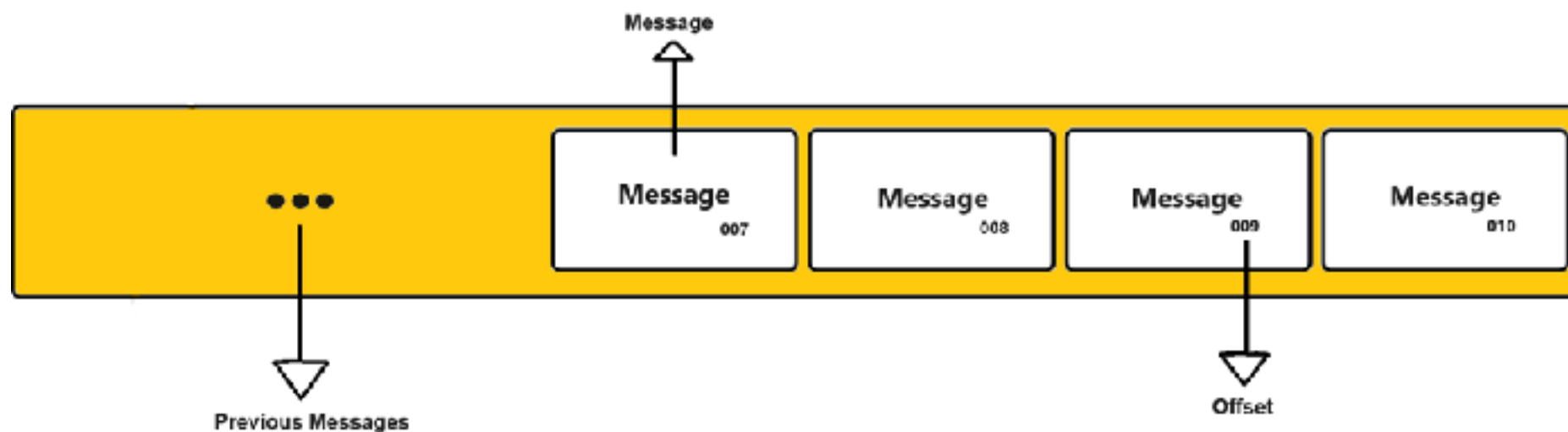
Stream

Design for high throughput

Real-time data stream

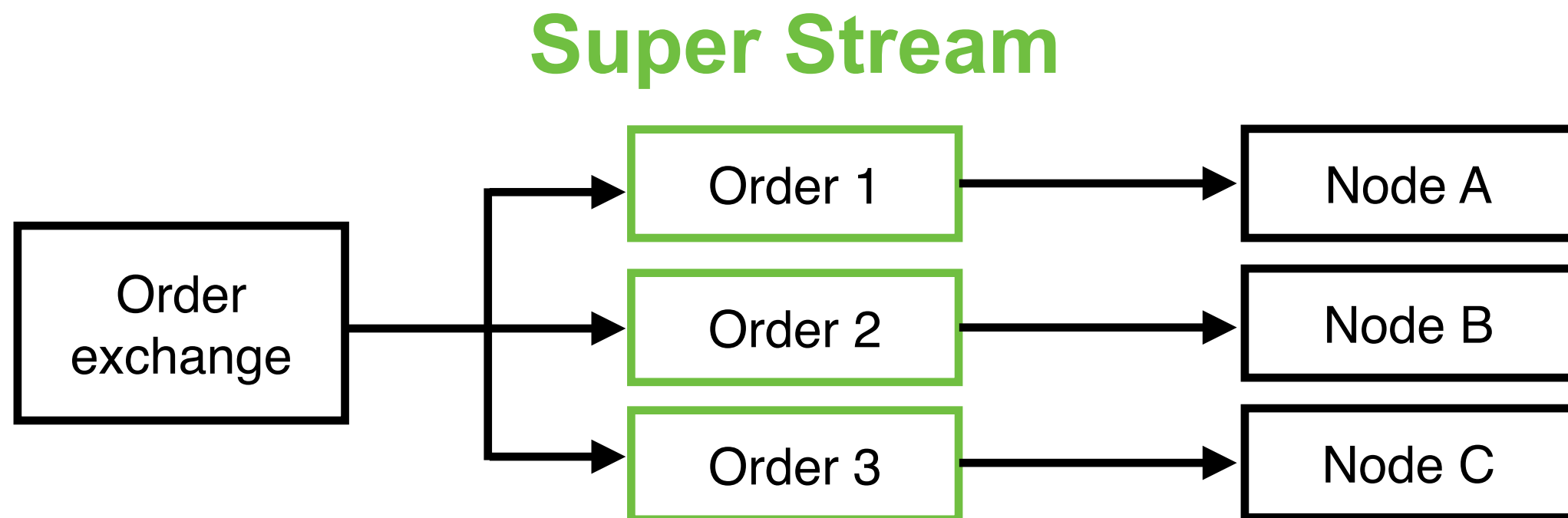
Message replay

Append-only log



Super Streams

Way to scale out by partitioning a **large stream** to small
Split storage and traffic on multiple nodes in cluster



<https://www.rabbitmq.com/blog/2022/07/13/rabbitmq-3-11-feature-preview-super-streams>



Working with Classic queue

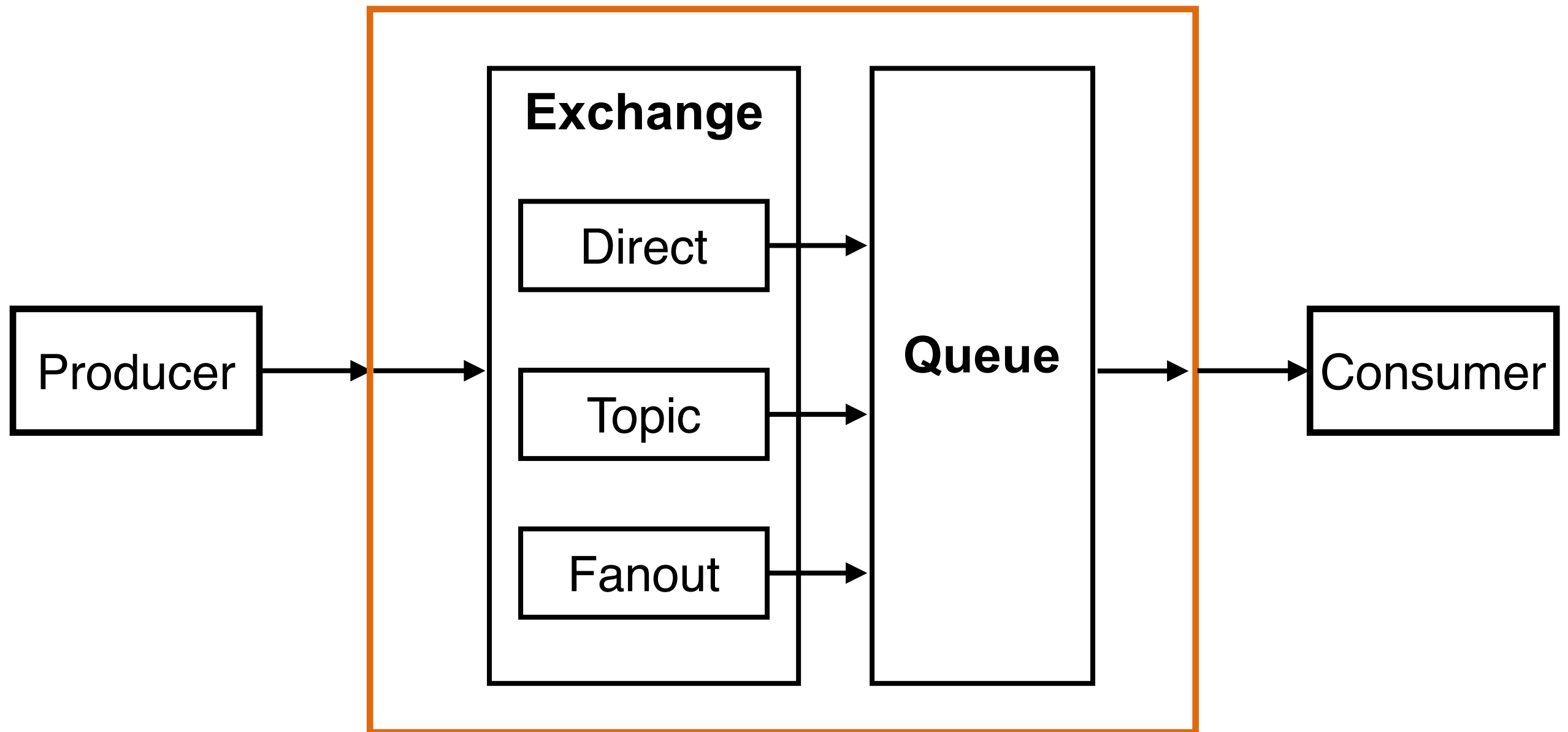


RabbitMQ works

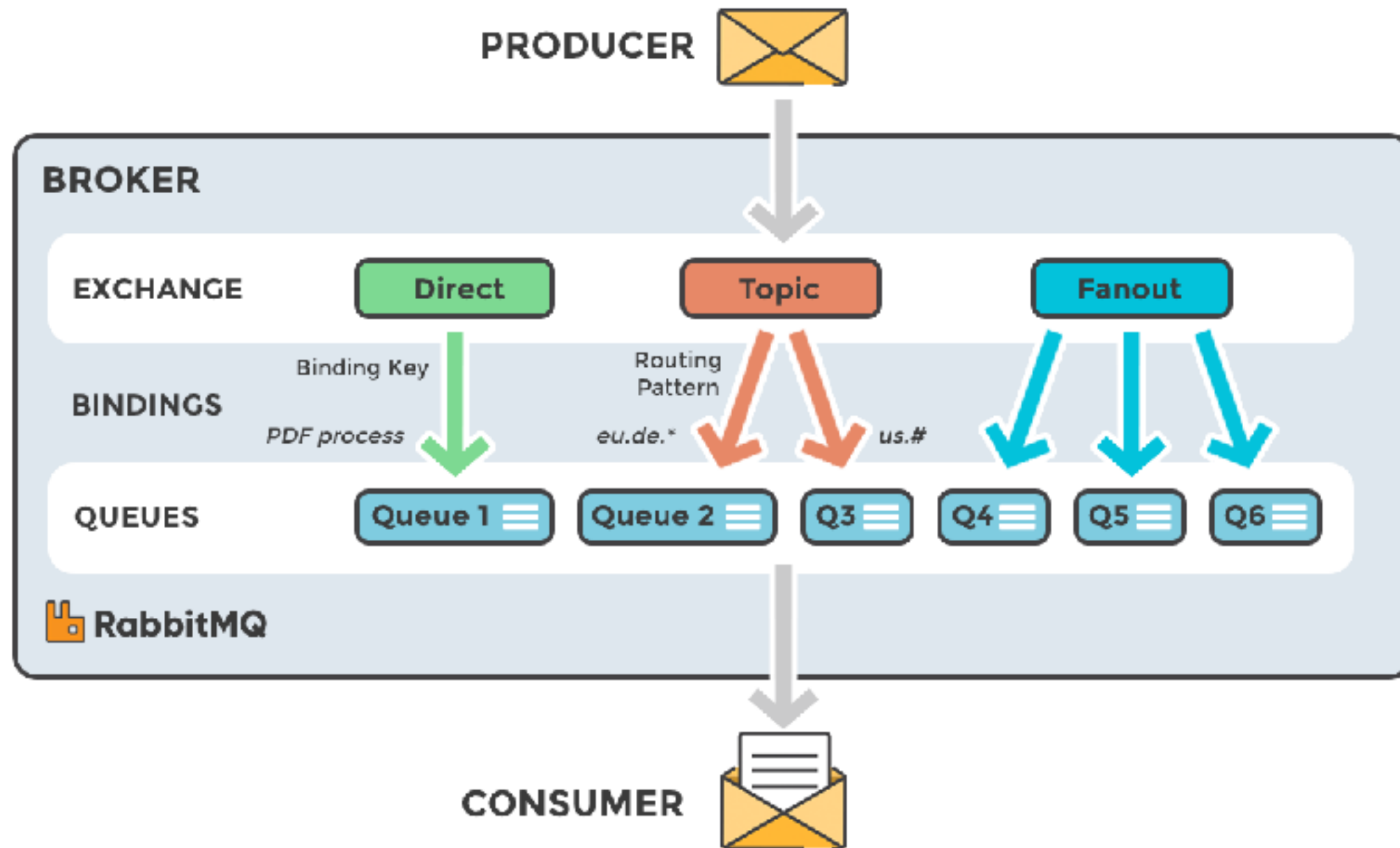


RabbitMQ works

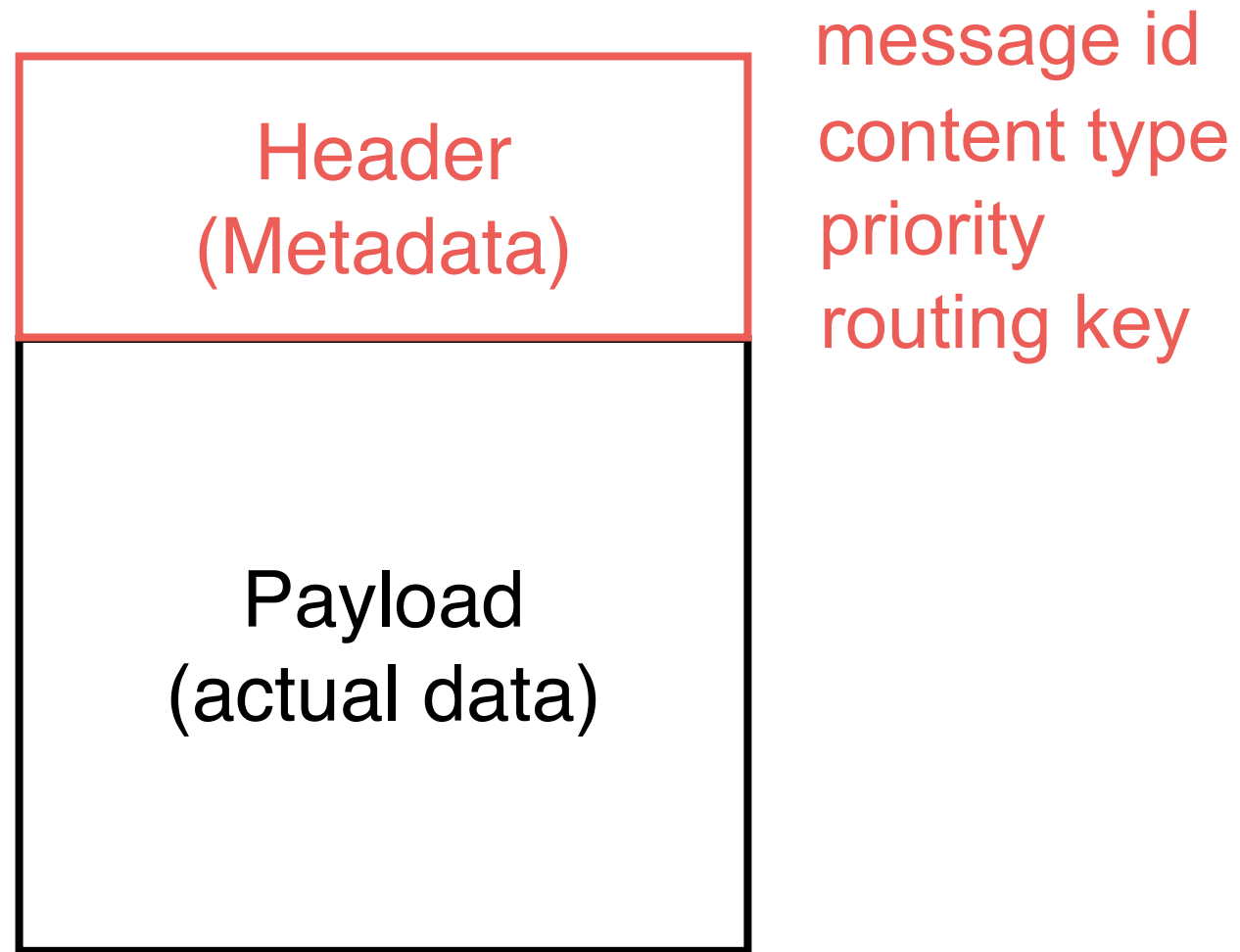
AMQP Broker



RabbitMQ works



Message structure



<https://www.rabbitmq.com/tutorials/amqp-concepts#messages>



AsyncAPI

Building the future of Event-Driven Architectures (EDA)

Open-Source tools to easily build and maintain your event-driven architecture. All powered by the AsyncAPI specification, the **industry standard** for defining asynchronous APIs.

[Read the docs](#) >

K

Proud to be part of the [Linux Foundation](#)

<https://www.asyncapi.com/en>



Exchange ?

Message routing agent within RabbitMQ

Receive message from producer and route to queue

Different exchange types handle routing key
in different way



Exchange types

Direct

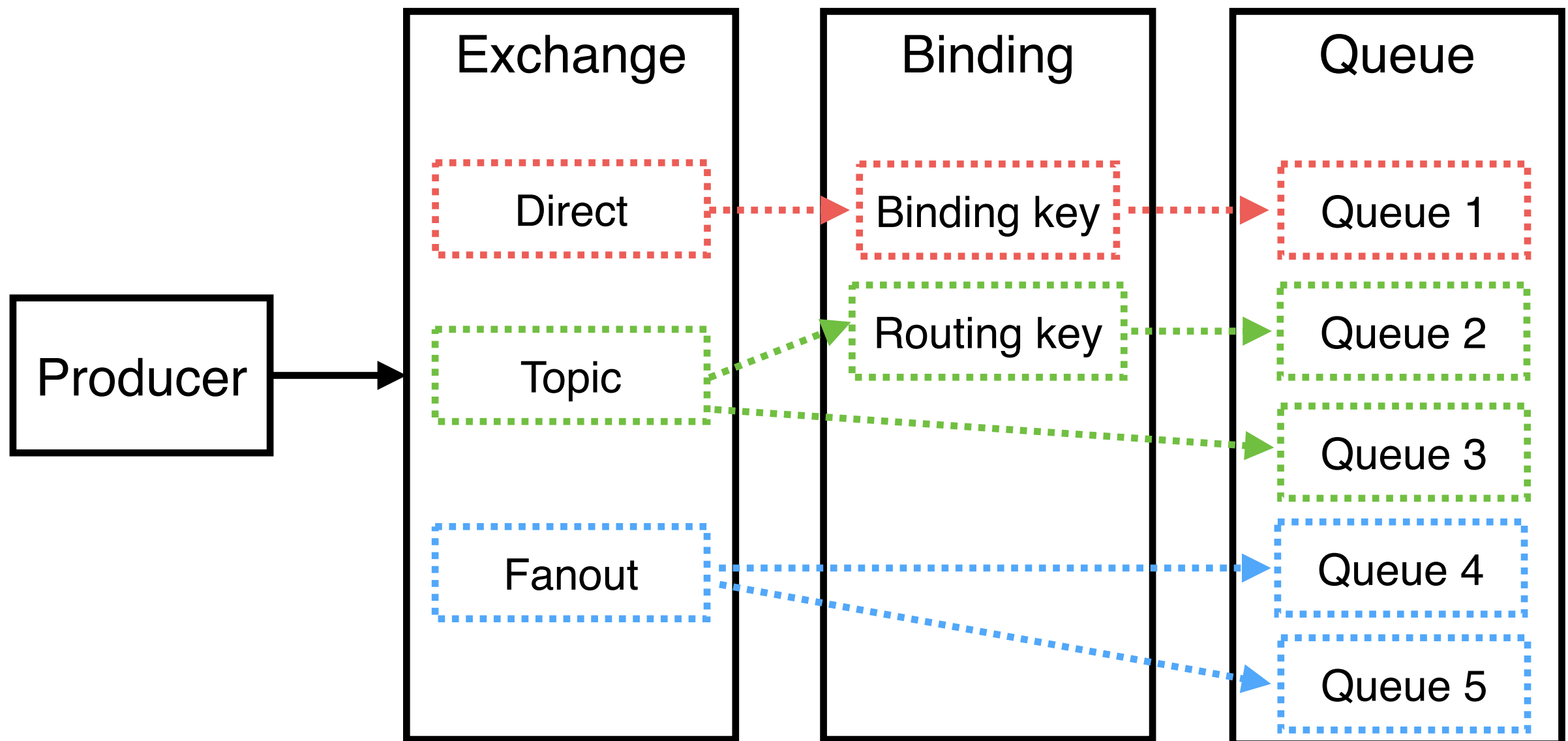
Fanout

Topic

Header



Exchange ?

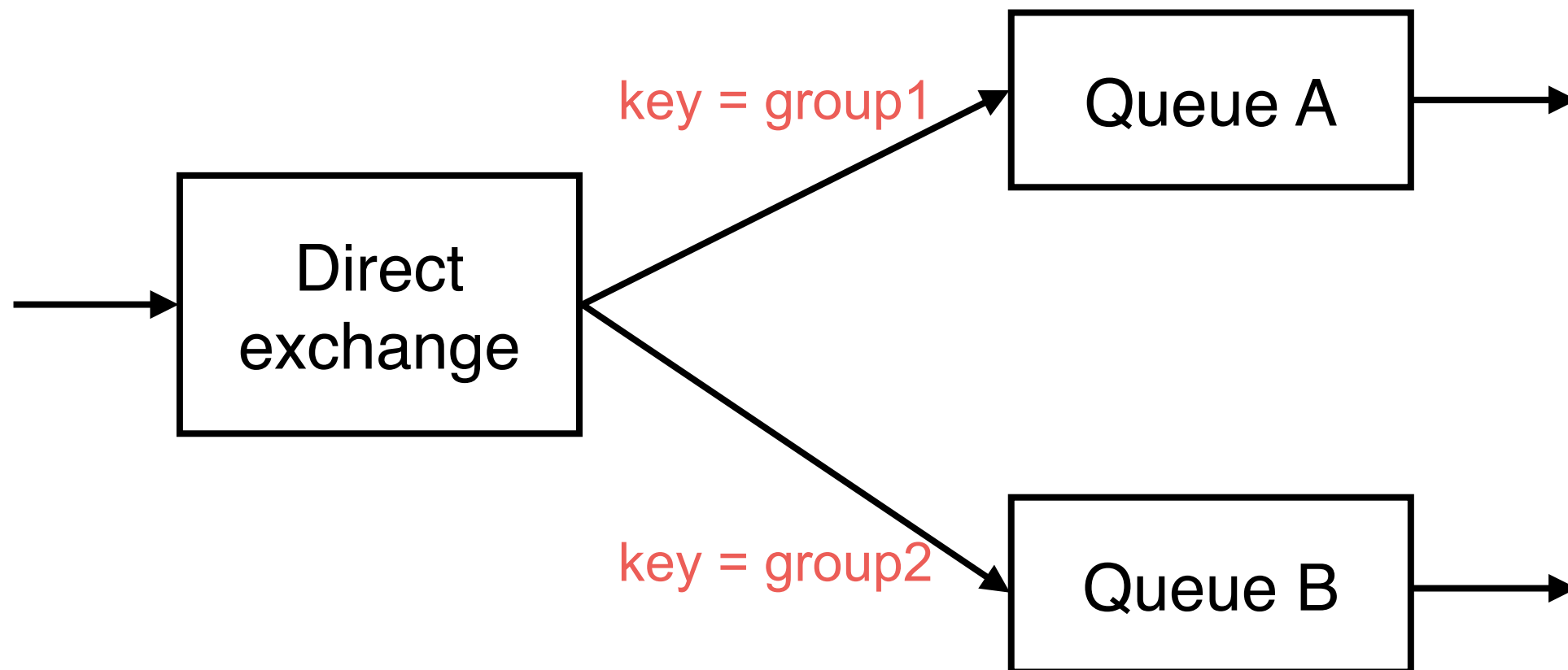


Direct Exchange



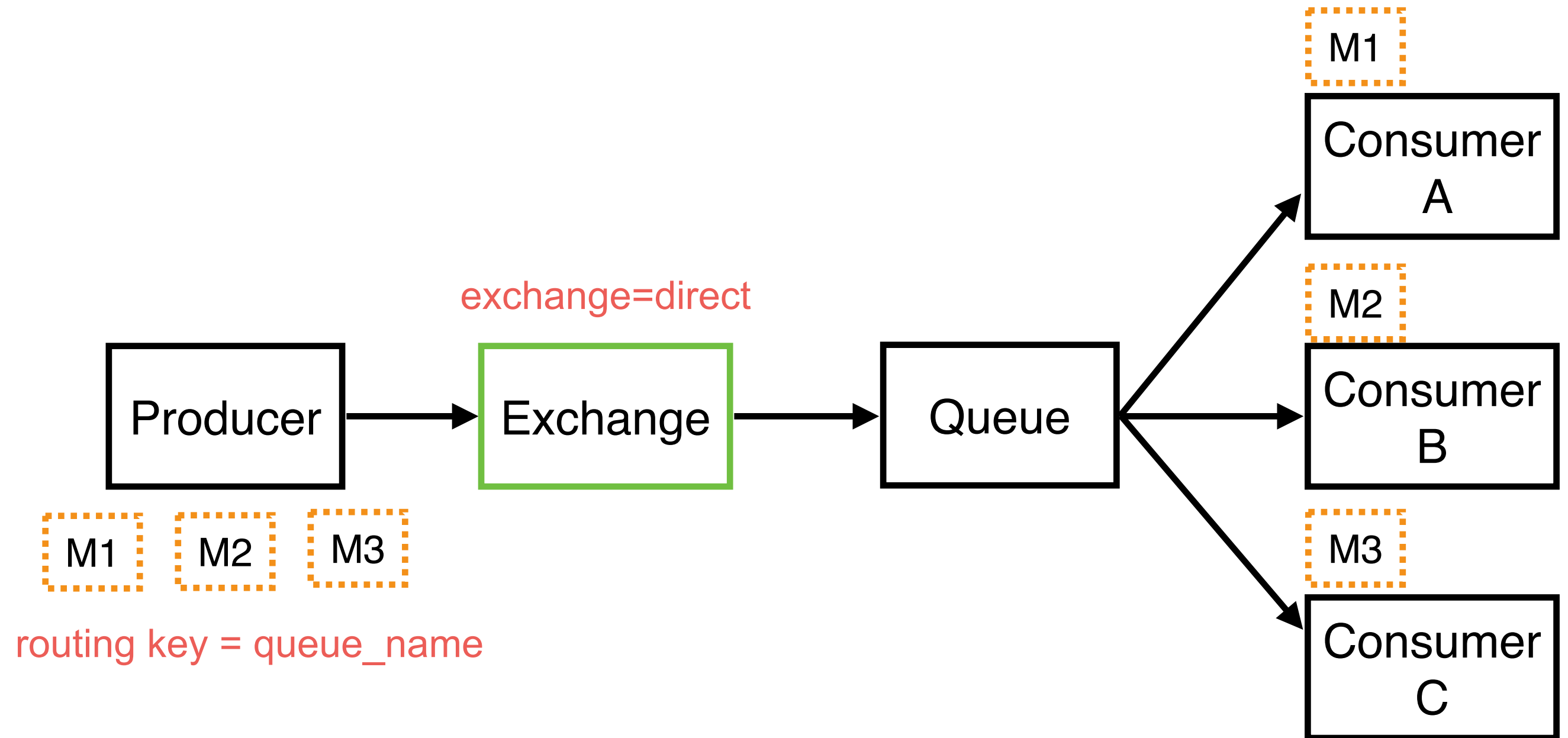
Direct exchange

Deliver message to queue based on **routing key**



1. Work Queues

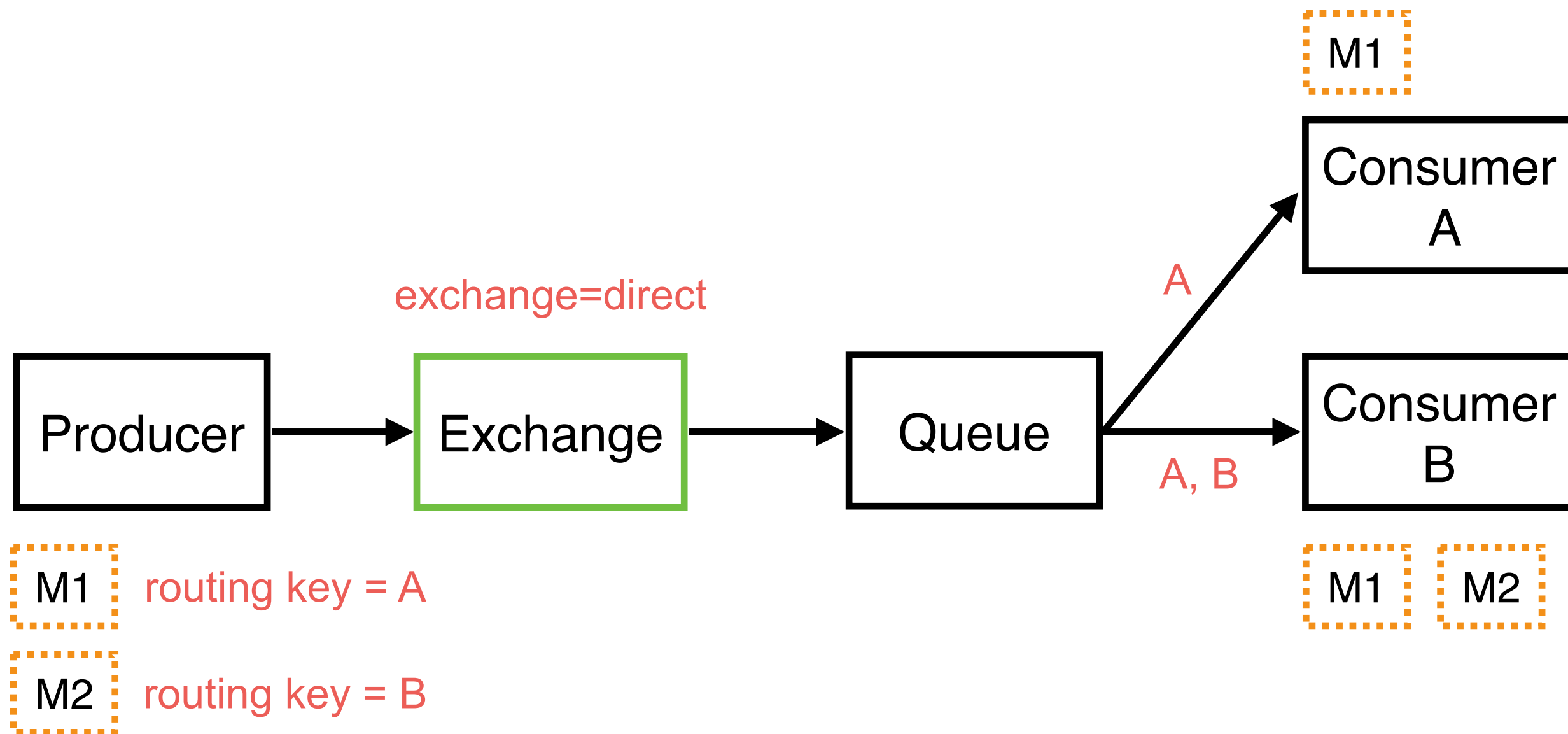
Exchange = empty = direct



<https://github.com/up1/course-rabbitmq-2025/tree/main/workshop/demo-go/work-queue>



2. Routing by key

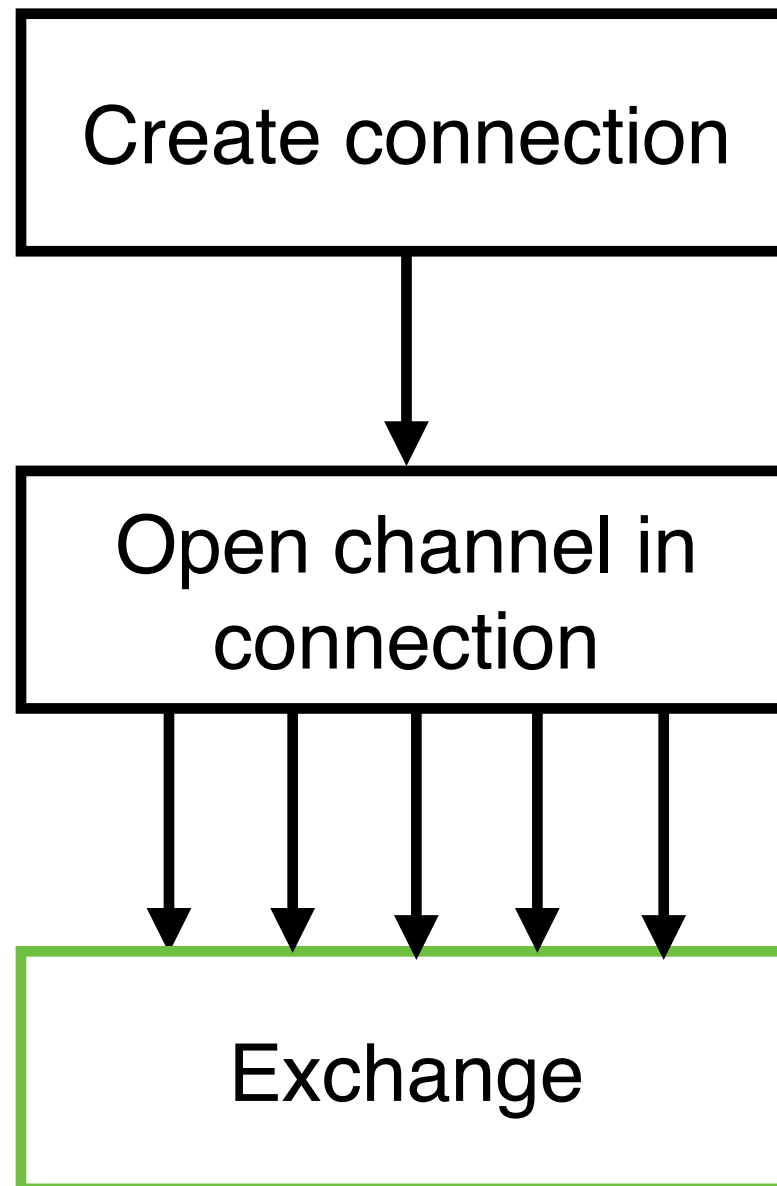


<https://github.com/up1/course-rabbitmq-2025/tree/main/workshop/demo-go/route-by-key>



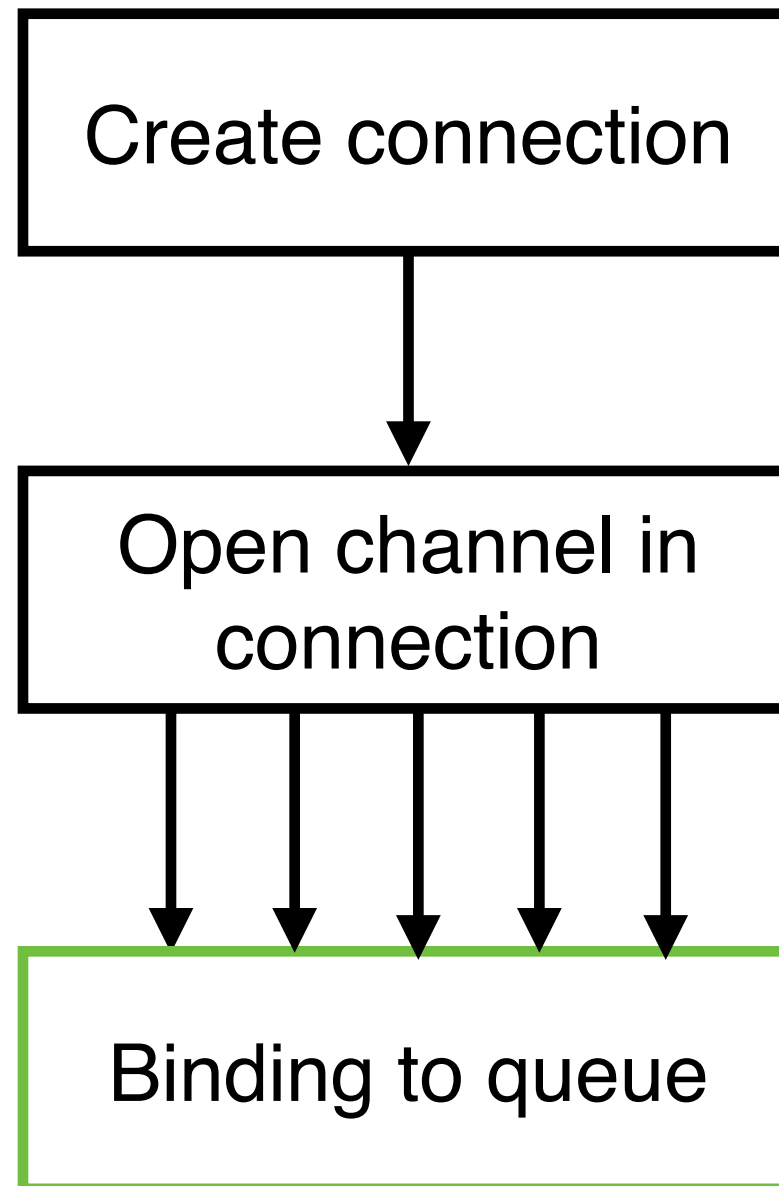
Producer

Step to publish message to exchange



Consumer

Step to read message from queue

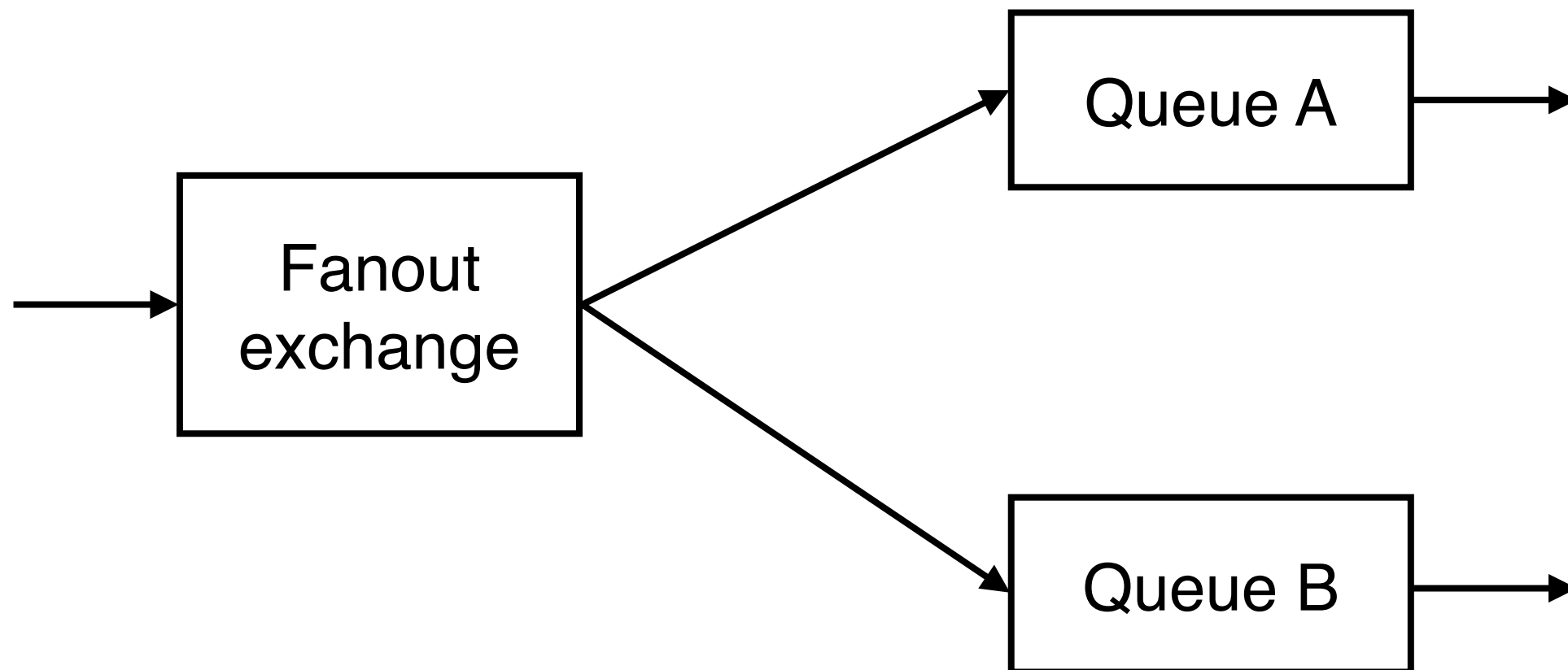


Fanout Exchange



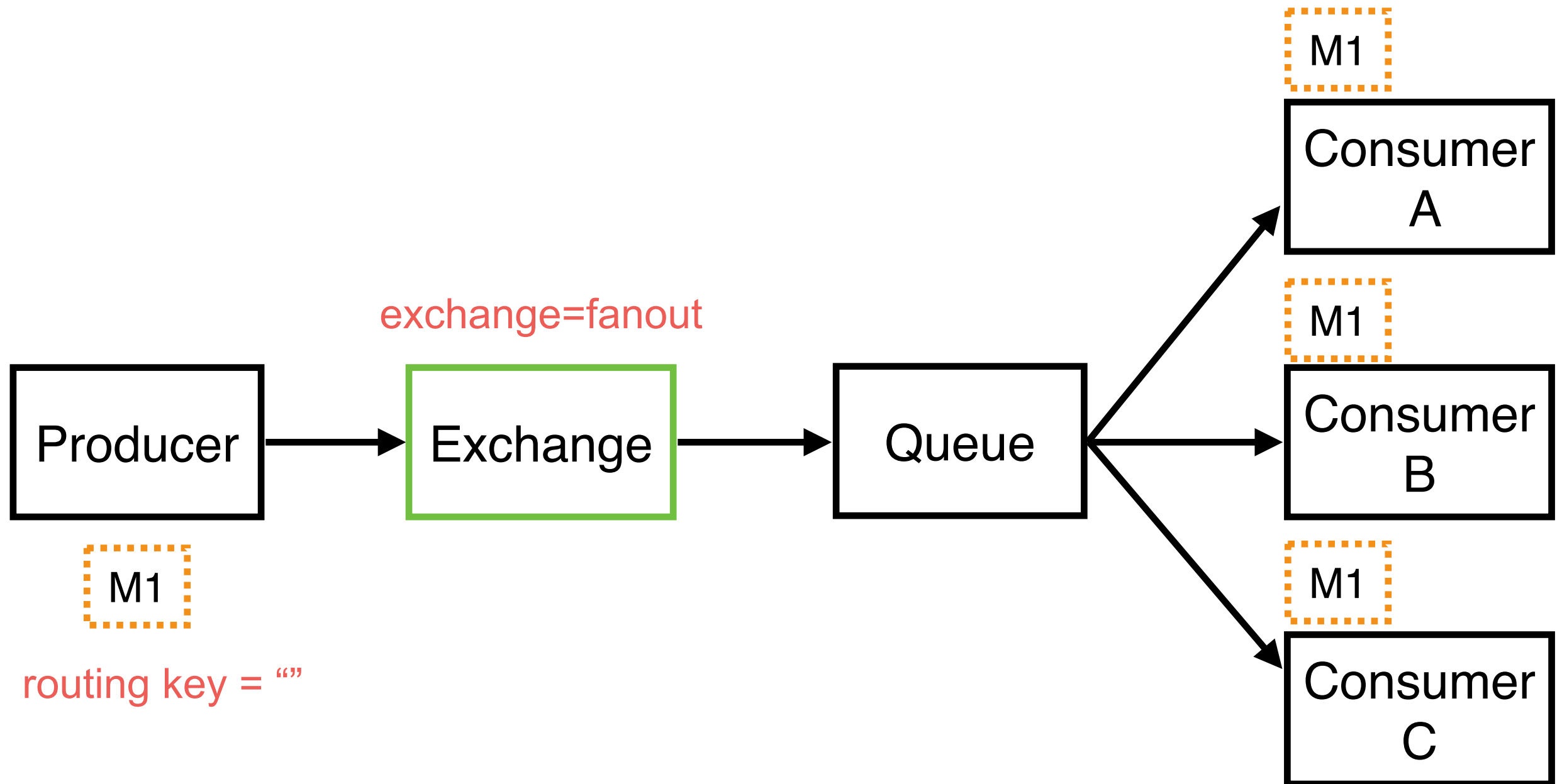
Fanout exchange

Broadcast messages to all bound queues
Ignore header, routing key in message



Publish/Subscribe

Exchange = fanout



<https://github.com/up1/course-rabbitmq-2025/tree/main/workshop/demo-go/pubsub>



Topic Exchange

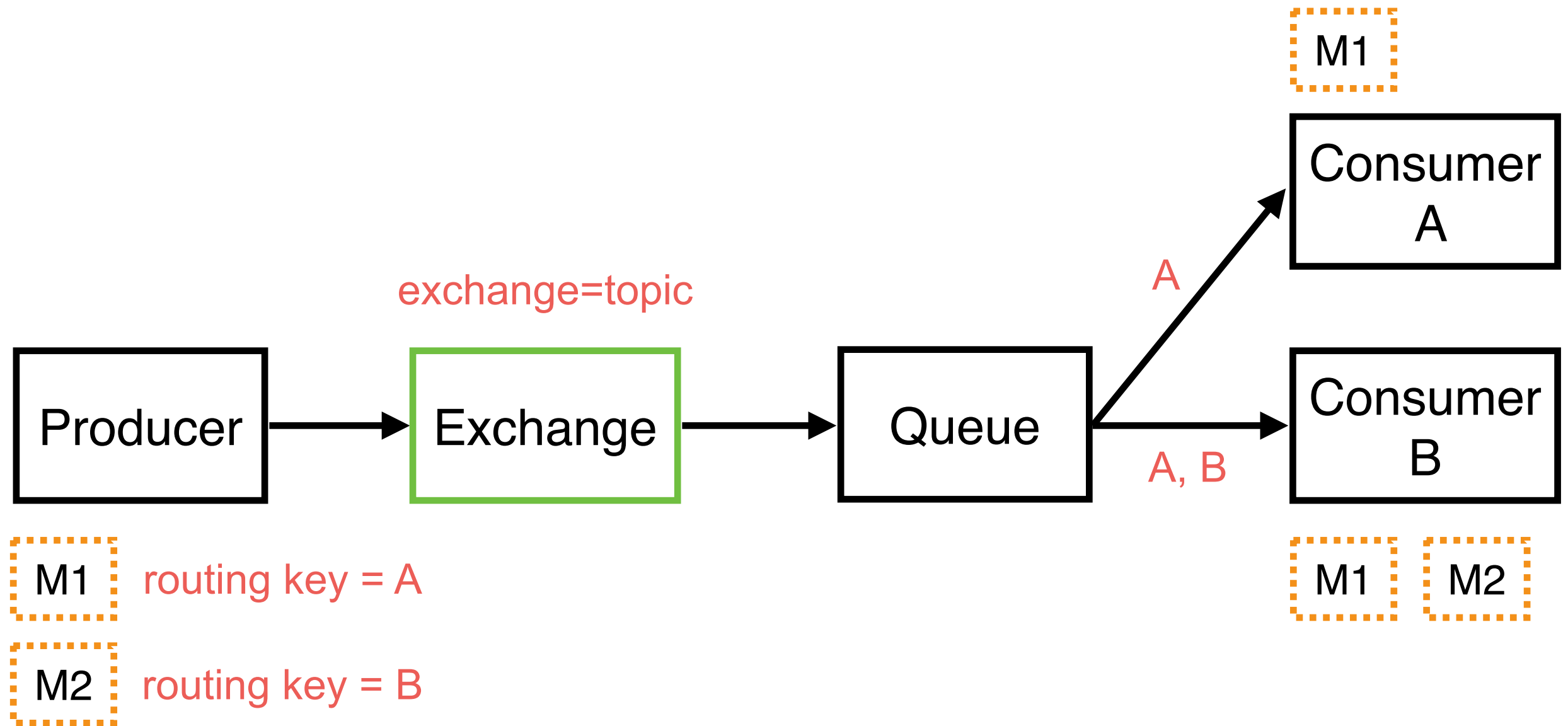


Topic exchange

Enables pattern-based routing
Using wildcard *, # in routing keys
Routing on multiple criteria



Routing by key*



<https://github.com/up1/course-rabbitmq-2025/tree/main/workshop/demo-go/topic>



Header Exchange



Header exchange

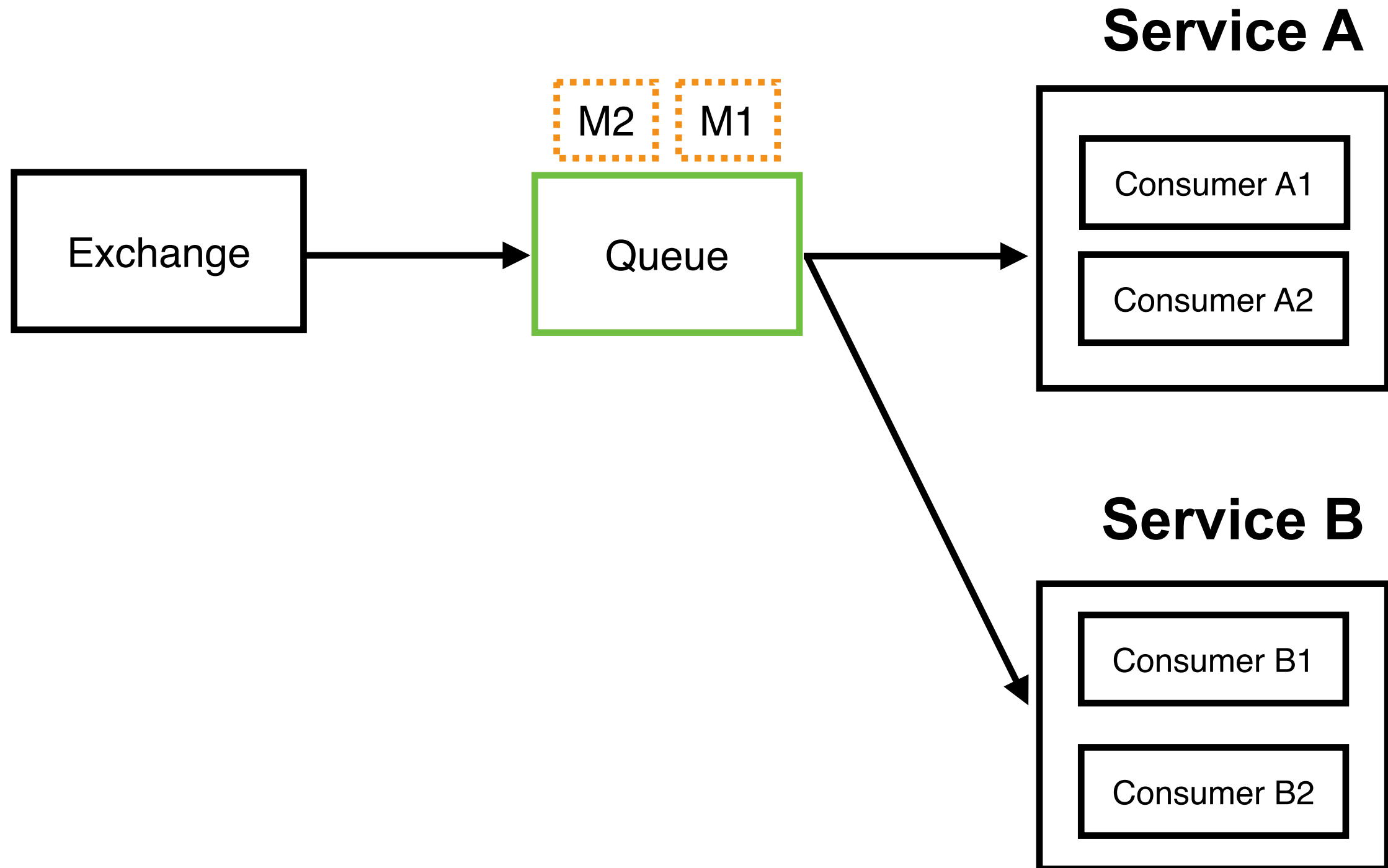
Using message's header to routing
Criteria based on header



Use Cases



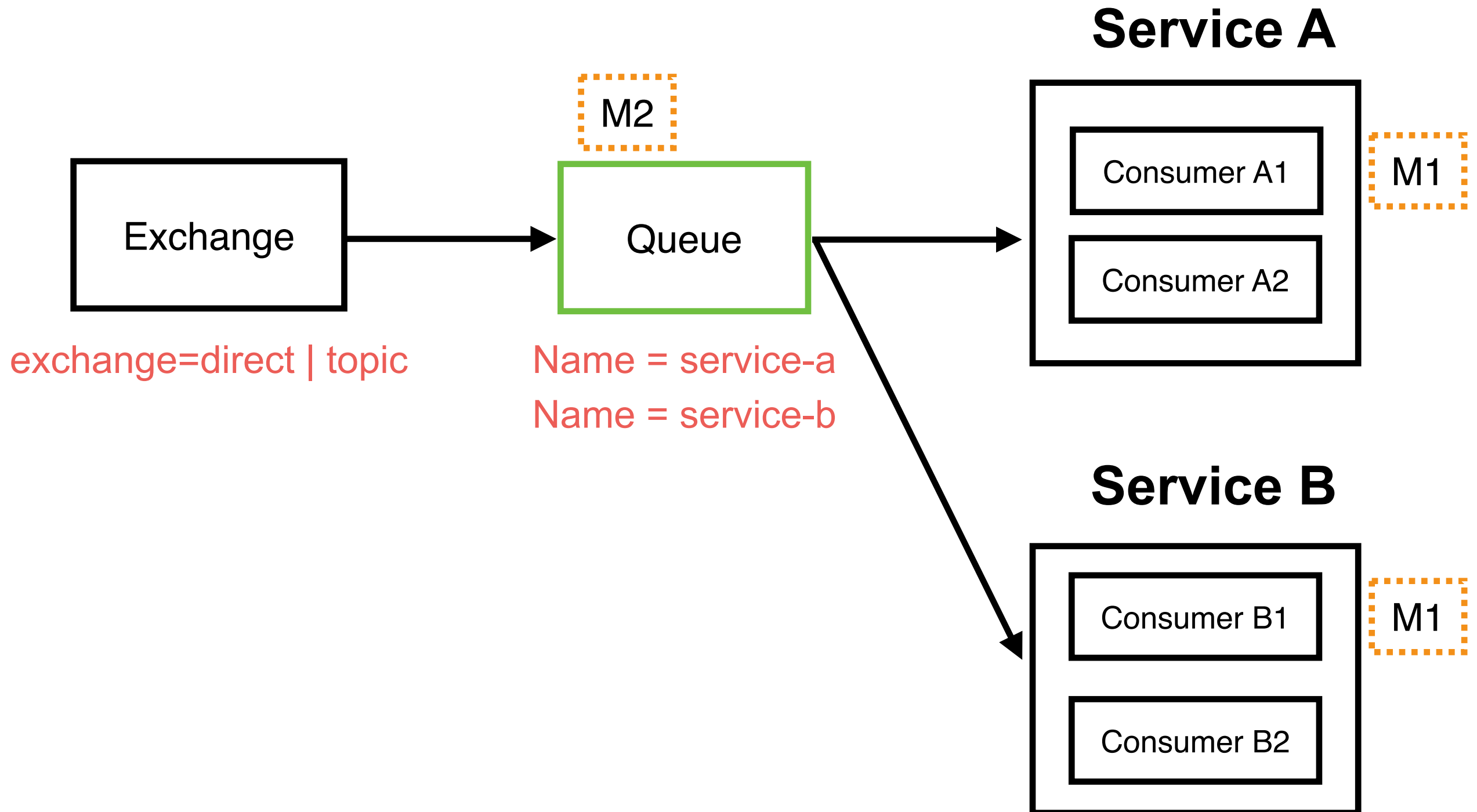
Scaling per services (1)



<https://github.com/up1/course-rabbitmq-2025/tree/main/workshop/demo-go/topic>



Scaling per services (2)



<https://github.com/up1/course-rabbitmq-2025/tree/main/workshop/demo-go/topic>



Failure Use Cases

Consumer crash (re-deliver)

Broker restart/crash

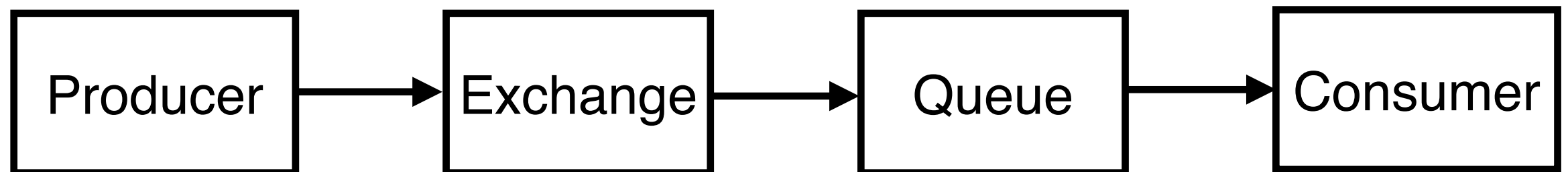
Retry with delay (N time)

Exceeding retry attempts

High-priority Dead Letter Queue (DLQ)



Consumer crash (re-deliver)



Delivery mode = Persistent

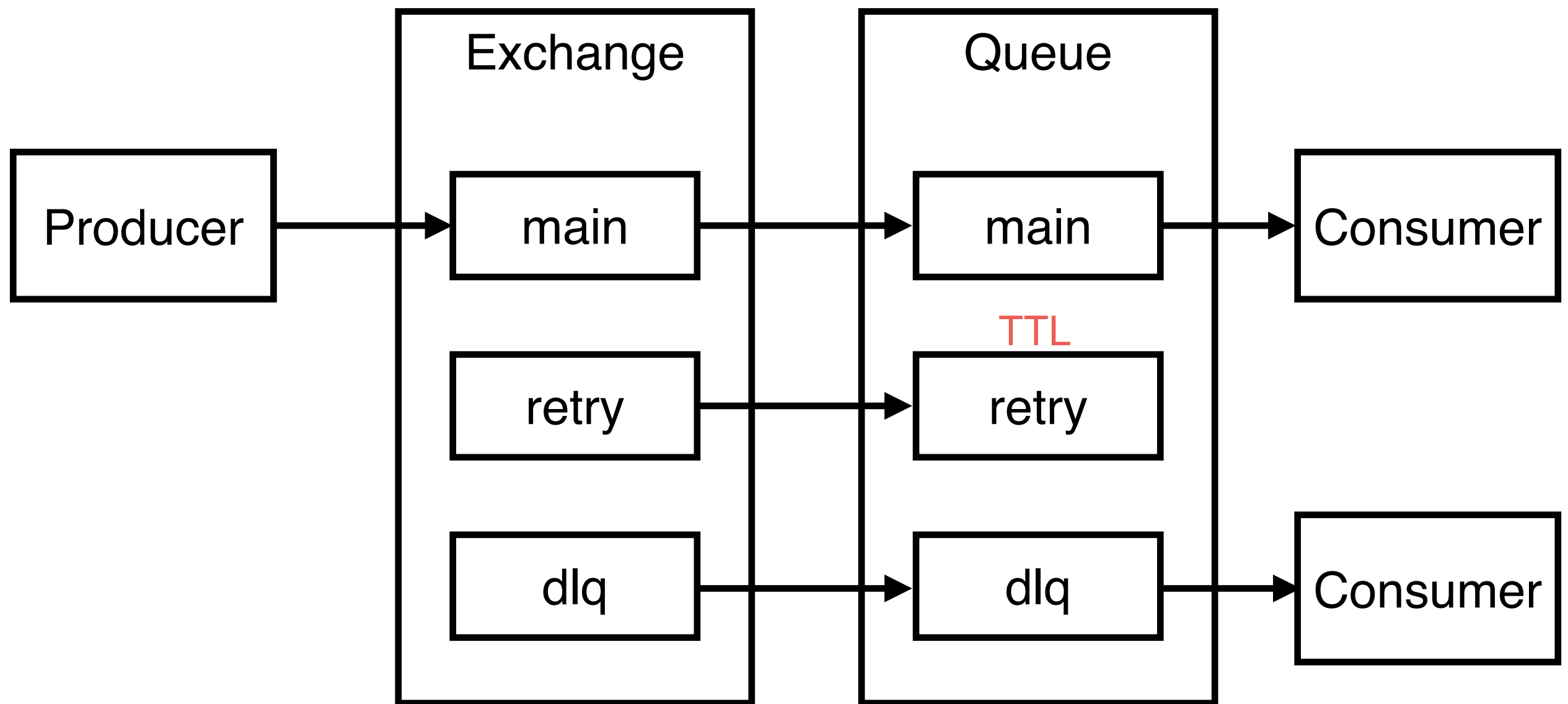
Durable = true
Delete = false

Auto ack = false

<https://github.com/up1/course-rabbitmq-2025/tree/main/workshop/demo-go/redeliver>



Retry and Dead Letter Queue

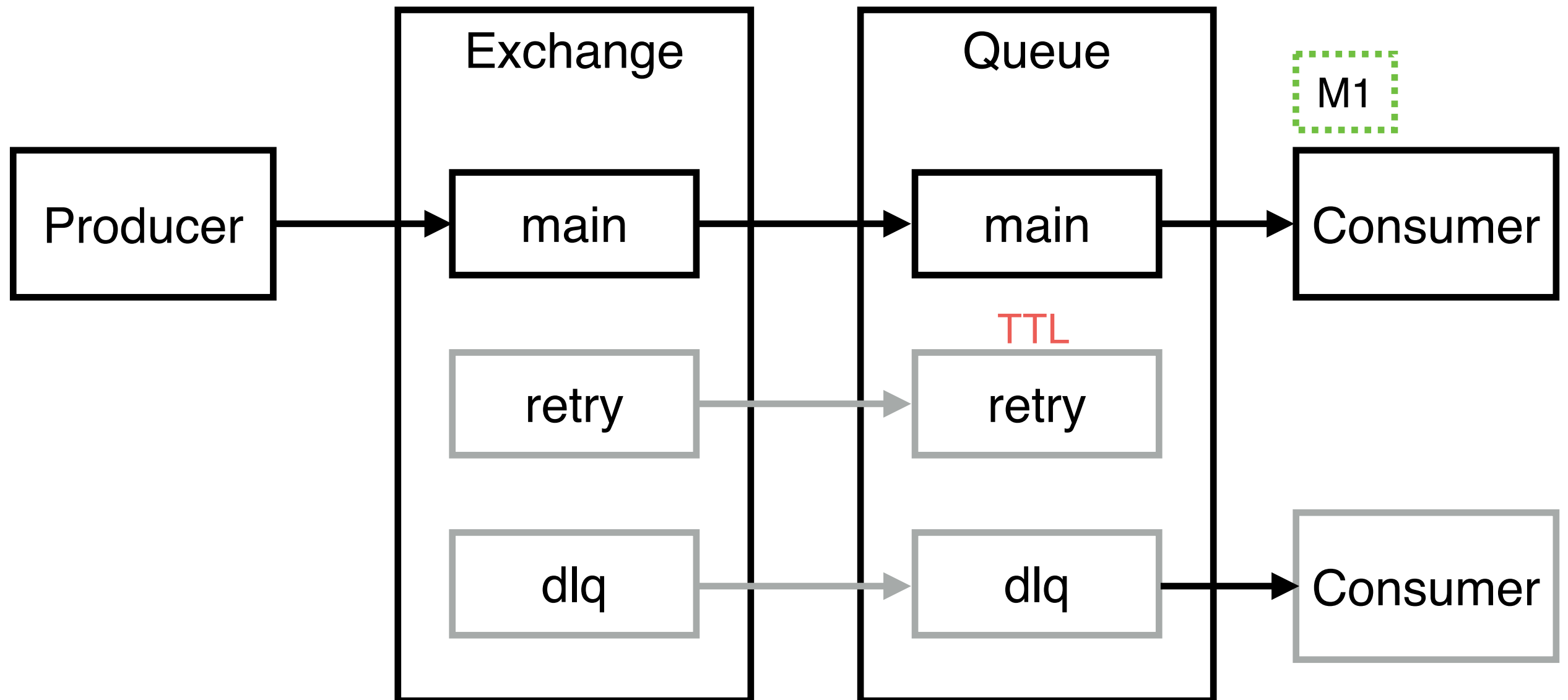


<https://github.com/up1/course-rabbitmq-2025/tree/main/workshop/demo-go/dlq>



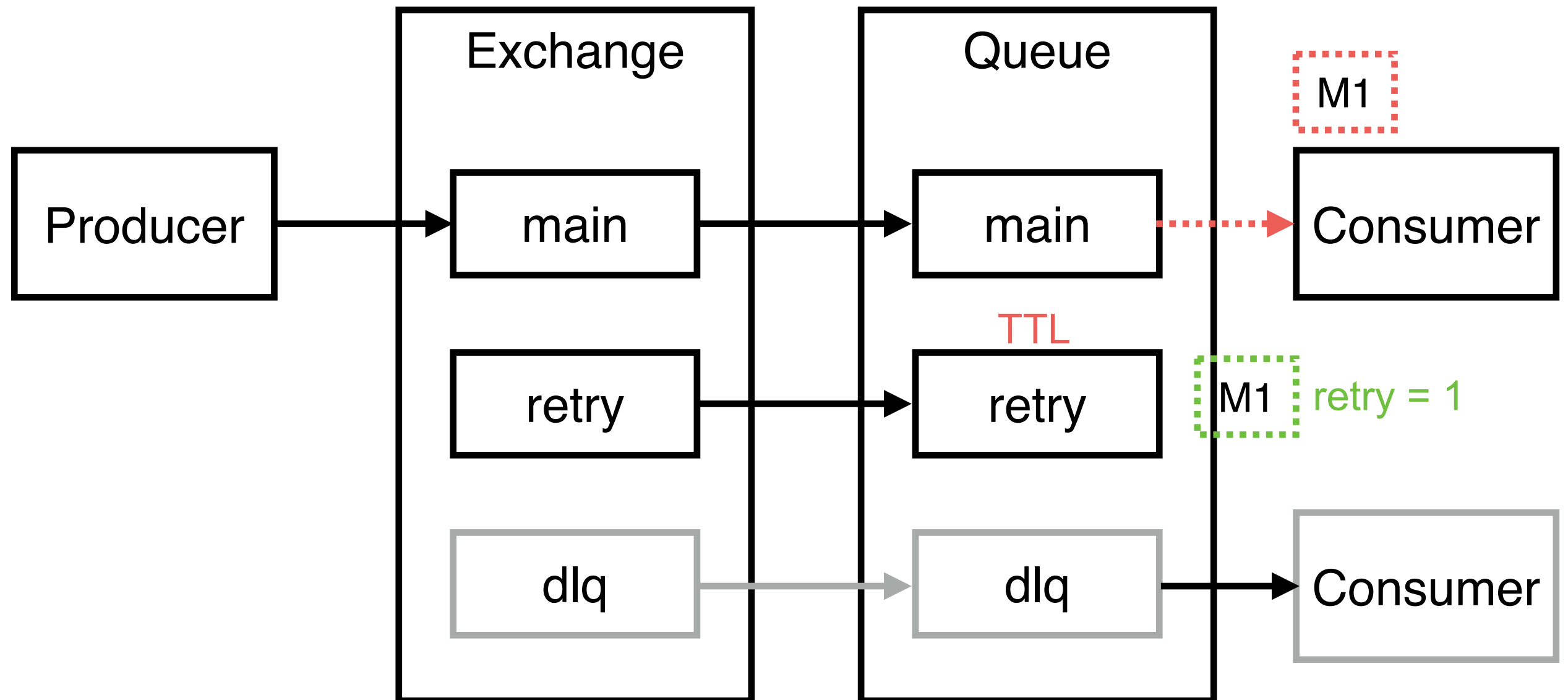
Retry and Dead Letter Queue

Normal process



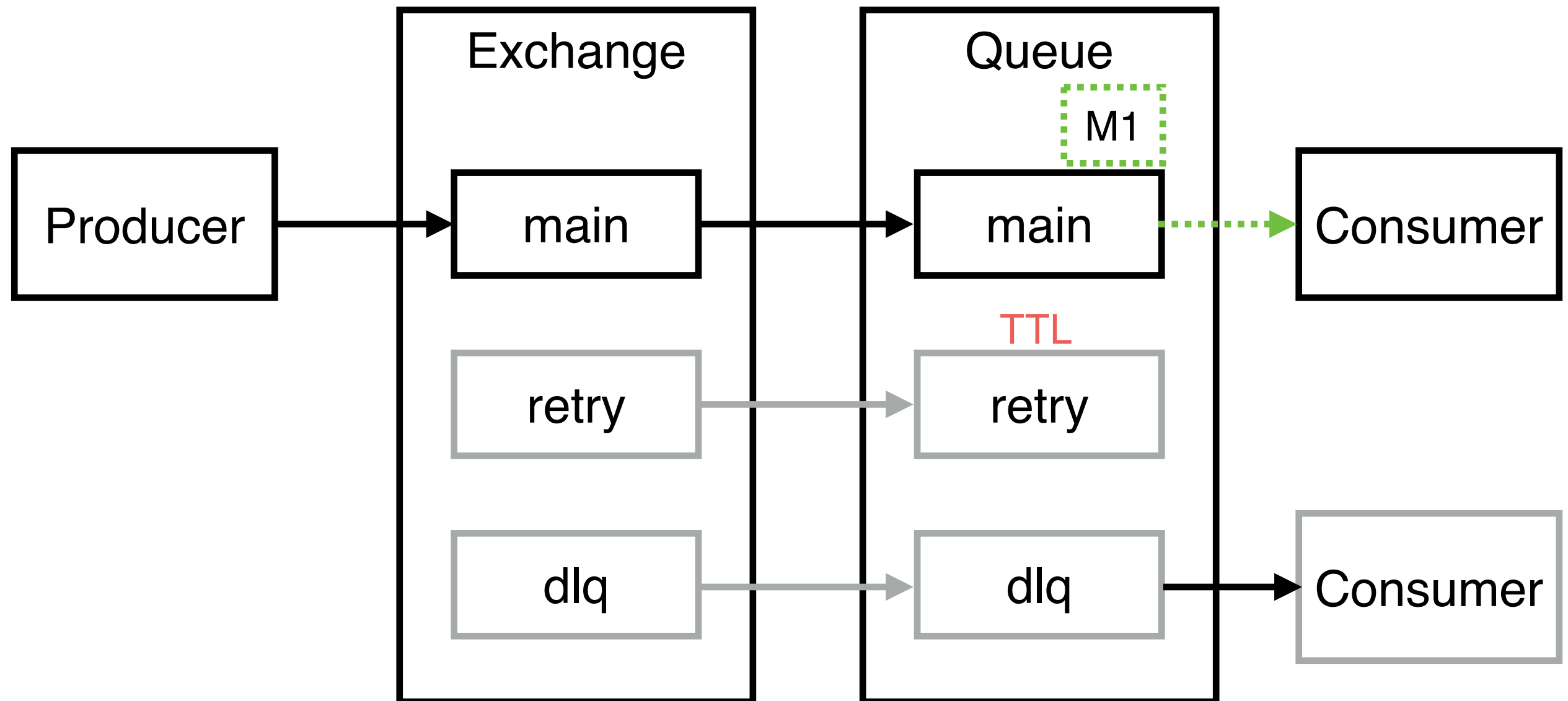
Retry and Dead Letter Queue

Problem in consumer, move message to **retry** exchange



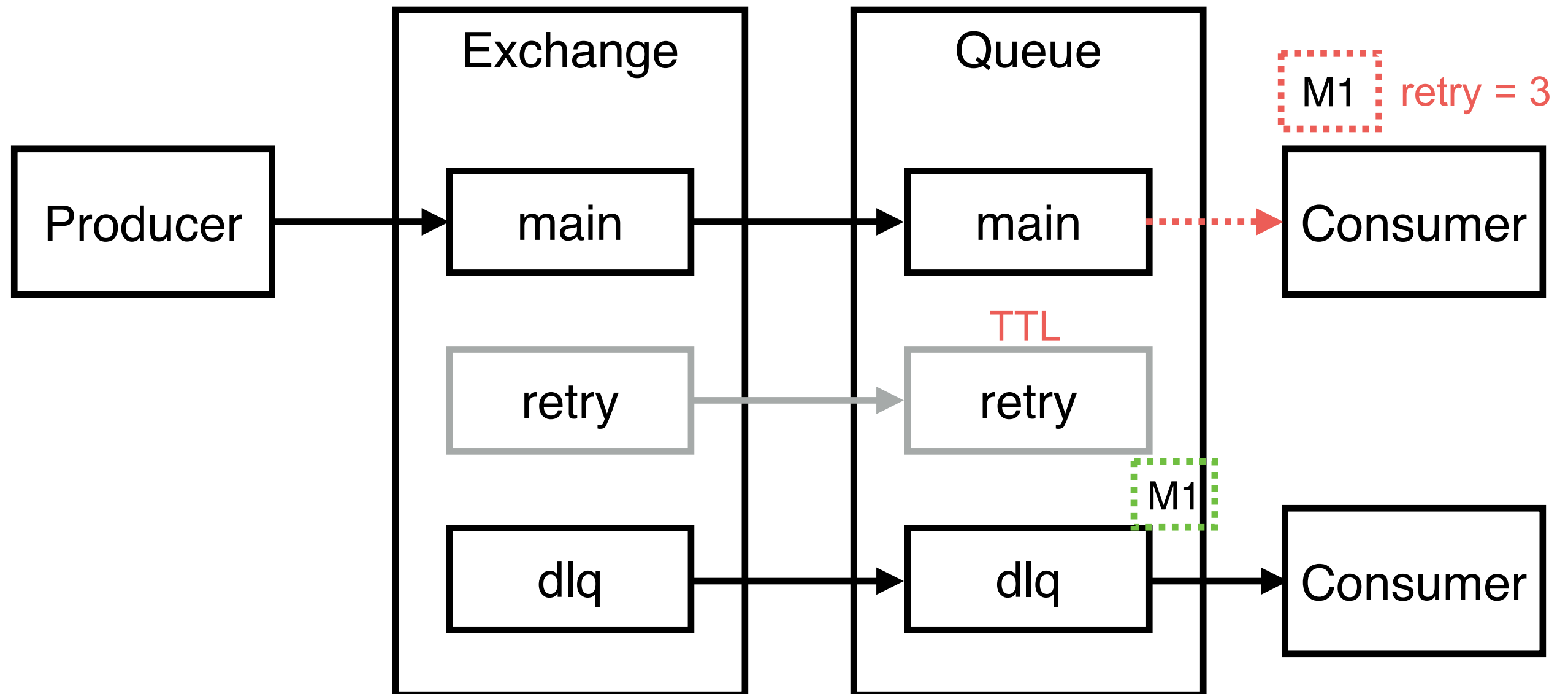
Retry and Dead Letter Queue

When TTL expired, move message to **main** exchange



Retry and Dead Letter Queue

When $\text{retry} > 3$ then move message to **dlq** exchange



RabbitMQ Stream

<https://www.rabbitmq.com/docs/streams>



Stream ?

Always persistent and replicated
Stream model an append-only log of message
Repeatedly read until message expire

Performance

High
Throughput

Low Latency



Key features

Append-only log
Non-destructive consumer semantics
Persistent and replicated
Retention policies (size-based, time-based)



Use Cases of Stream ?

Large fan-out

Replay

Large volume of
message

High Throughput

Event sourcing

Time-series data



Resources usage ?

All data is store on **disk**

Use disk I/O heavy

Lower CPU and memory than quorum

Tuning kernel page cache



<https://www.rabbitmq.com/docs/streams#resource-use>



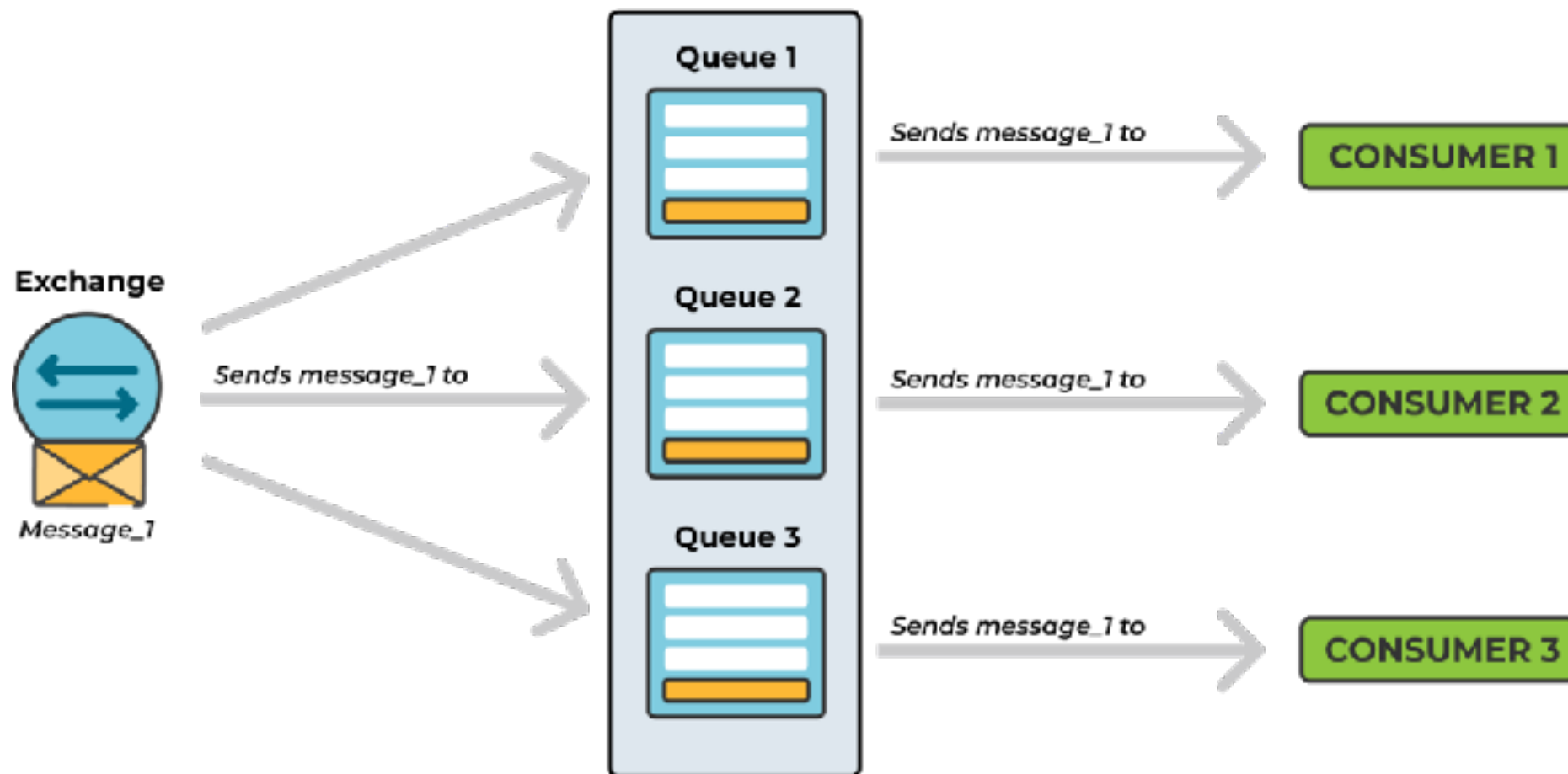
Queue vs Stream

| Feature | Queue | Stream |
|------------------|---|---|
| Message handling | FIFO destructive read | Append-only log non-destructive read |
| Persistence | Optional | Always persistent and replicated |
| Retention | TTL (Time To Live) Queue length limits | Size-based, time-based |
| Performance | Optimize for point-to-point | Optimize for large fanout and replay |



Fanout with Queue ?

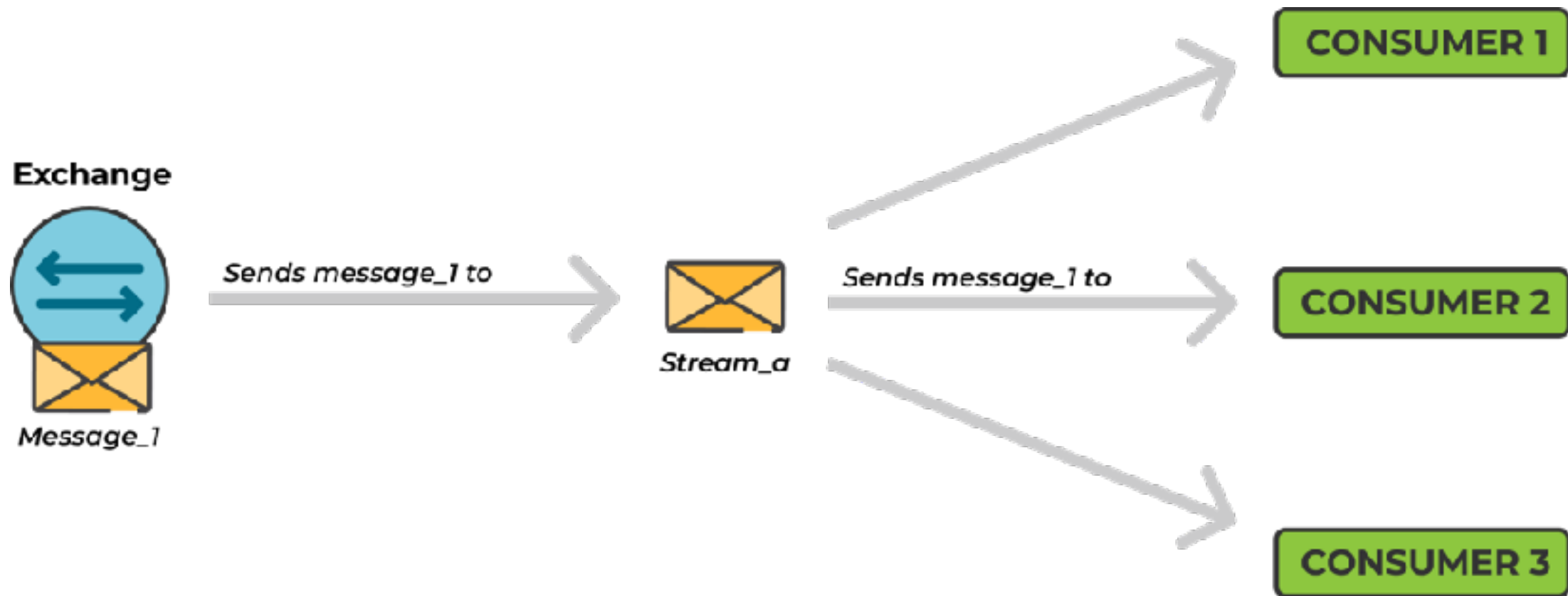
Multiple consumers read the same message



<https://www.cloudamqp.com/blog/rabbitmq-streams-and-replay-features-part-1-when-to-use-rabbitmq-streams.html>



Fanout with Stream

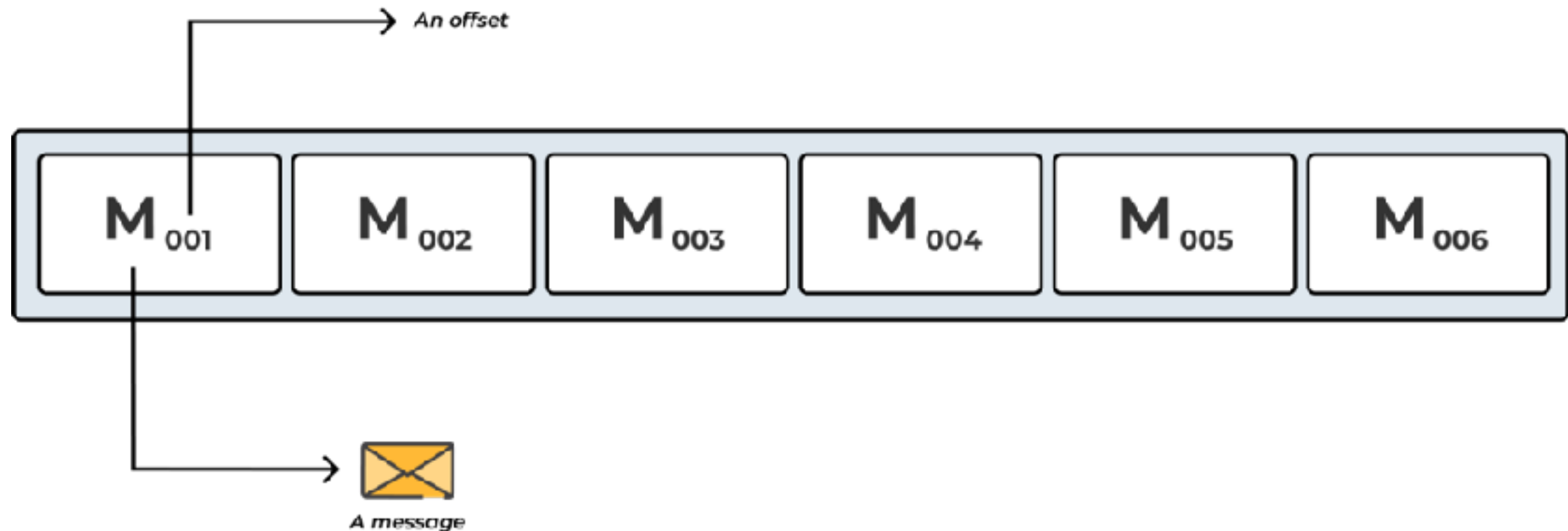


<https://www.cloudamqp.com/blog/rabbitmq-streams-and-replay-features-part-1-when-to-use-rabbitmq-streams.html>

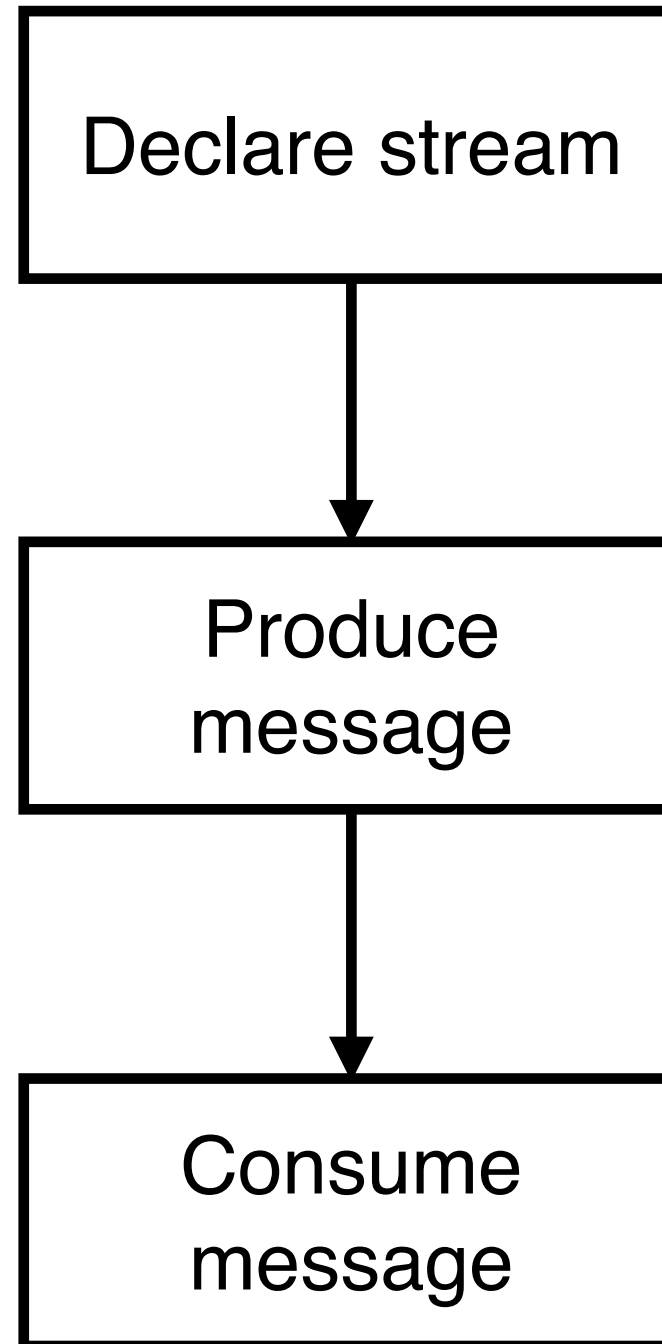


Replay with offset

When consumer needs to re-read the same message



Steps to use Stream ?



Workshop

Fanout with Stream

Fanout

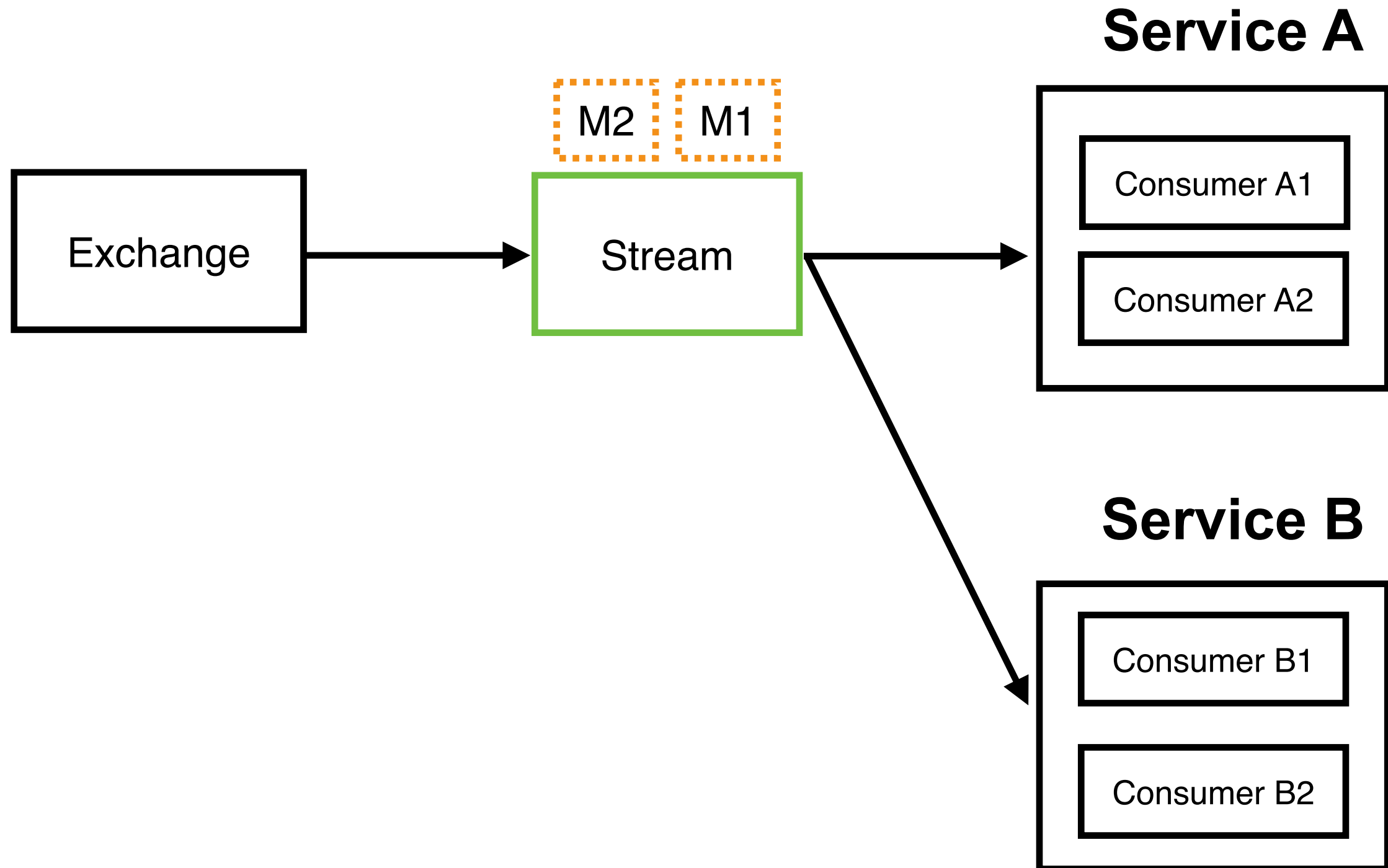
Tracking offset

Filter message

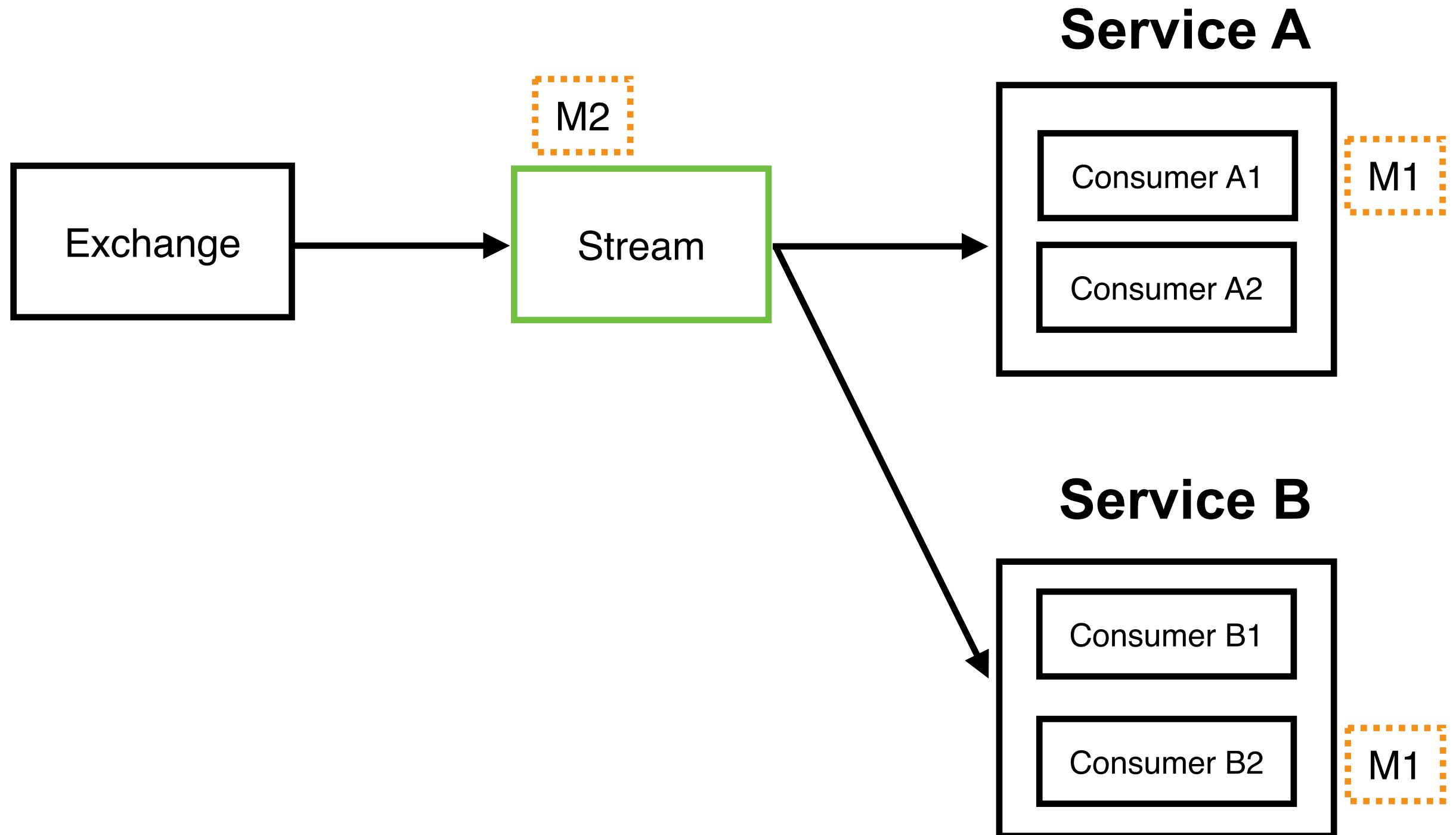
<https://github.com/up1/course-rabbitmq-2025/tree/main/workshop/demo-go/demo-stream/basic>



Requirement (1)



Requirement (2)



SAC

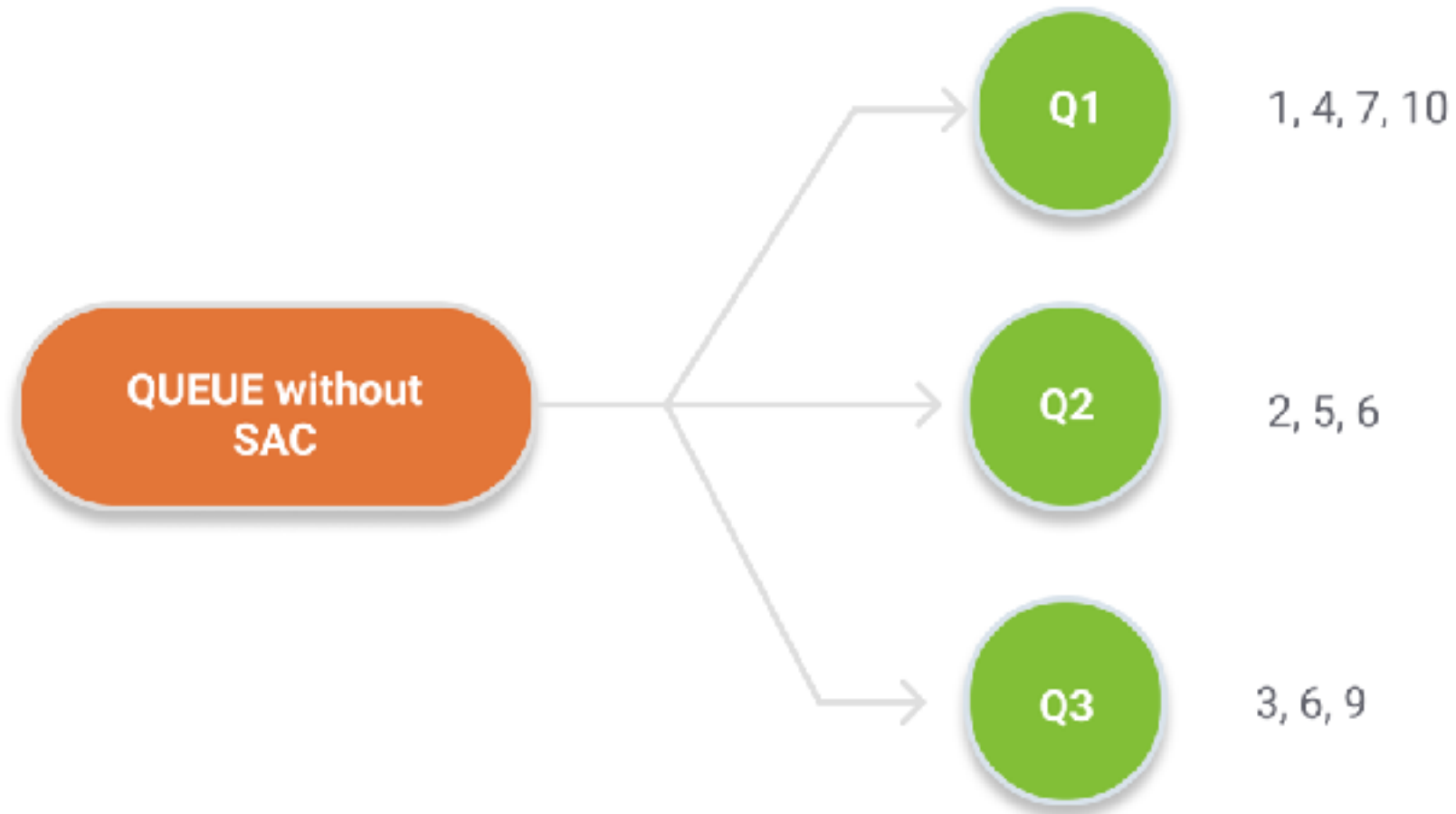
Single Active Consumer

<https://www.rabbitmq.com/docs/consumers#single-active-consumer>

<https://www.rabbitmq.com/docs/streams#single-active-consumer>



Queue without SAC



<https://www.cloudamqp.com/blog/rabbitmq-3-8-feature-focus-single-active-consumer.html>



Single Active Consumer

All messages get delivered to a **active consumer**
Auto switch to other consumers when active failed



<https://www.cloudamqp.com/blog/rabbitmq-3-8-feature-focus-single-active-consumer.html>



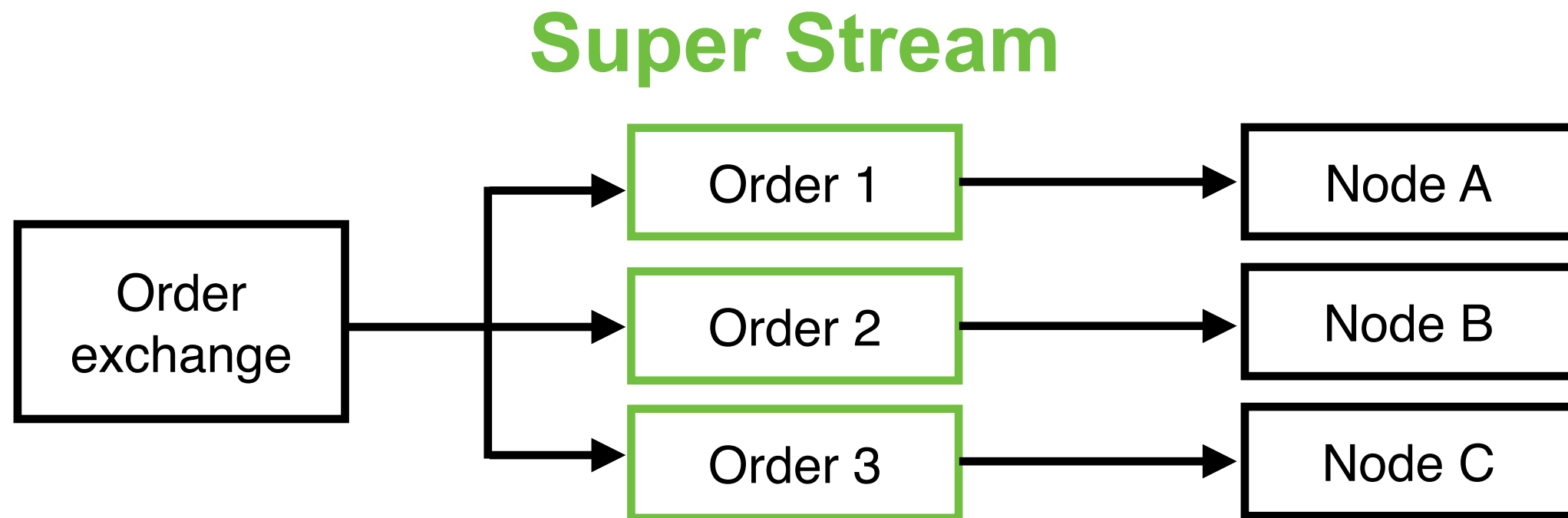
Super Streams

(Partition the stream)



Super Streams ?

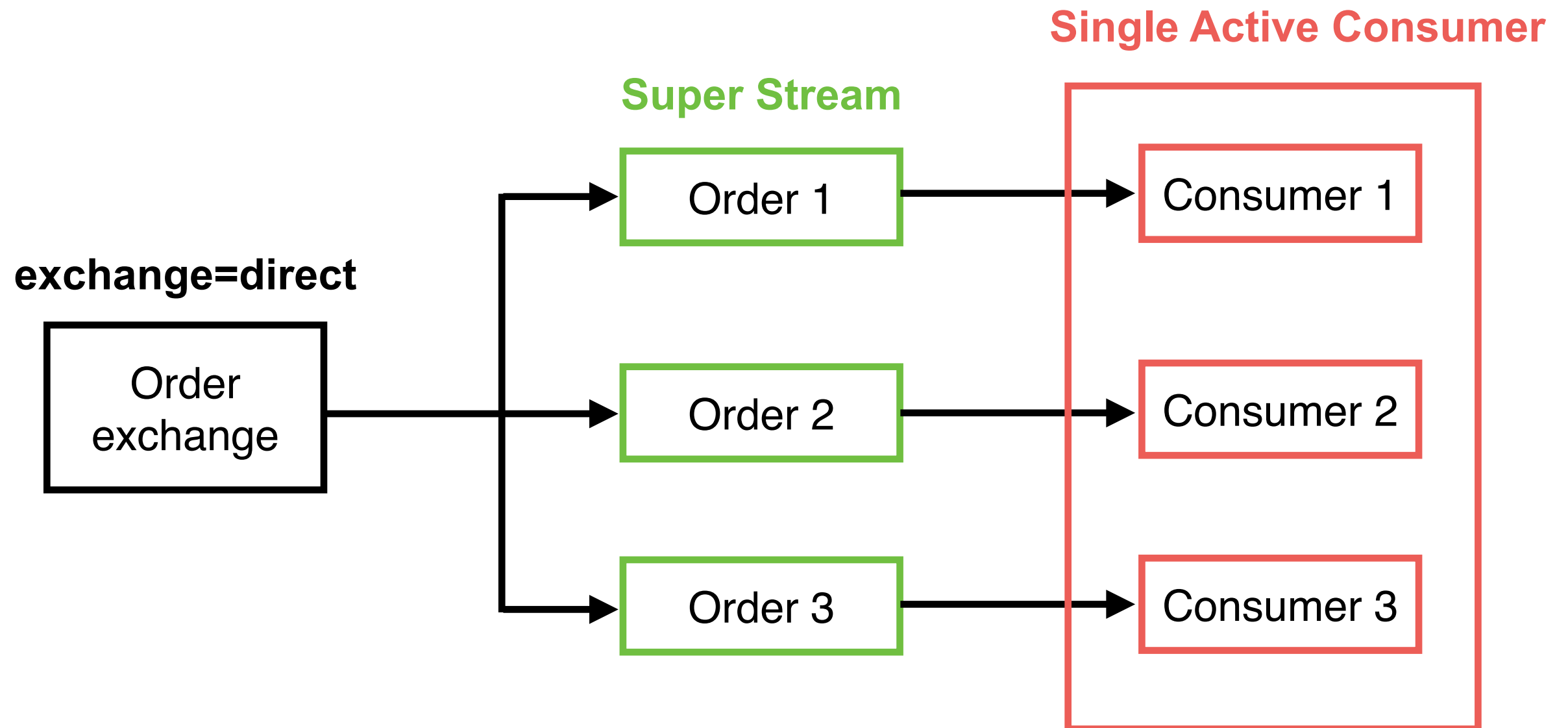
Way to scale out by partitioning a **large stream** to small
Split storage and traffic on multiple nodes in cluster



<https://www.rabbitmq.com/blog/2022/07/13/rabbitmq-3-11-feature-preview-super-streams>



Workshop with Orders



<https://github.com/up1/course-rabbitmq-2025/tree/main/workshop/demo-go/demo-stream/basic>



Q/A

