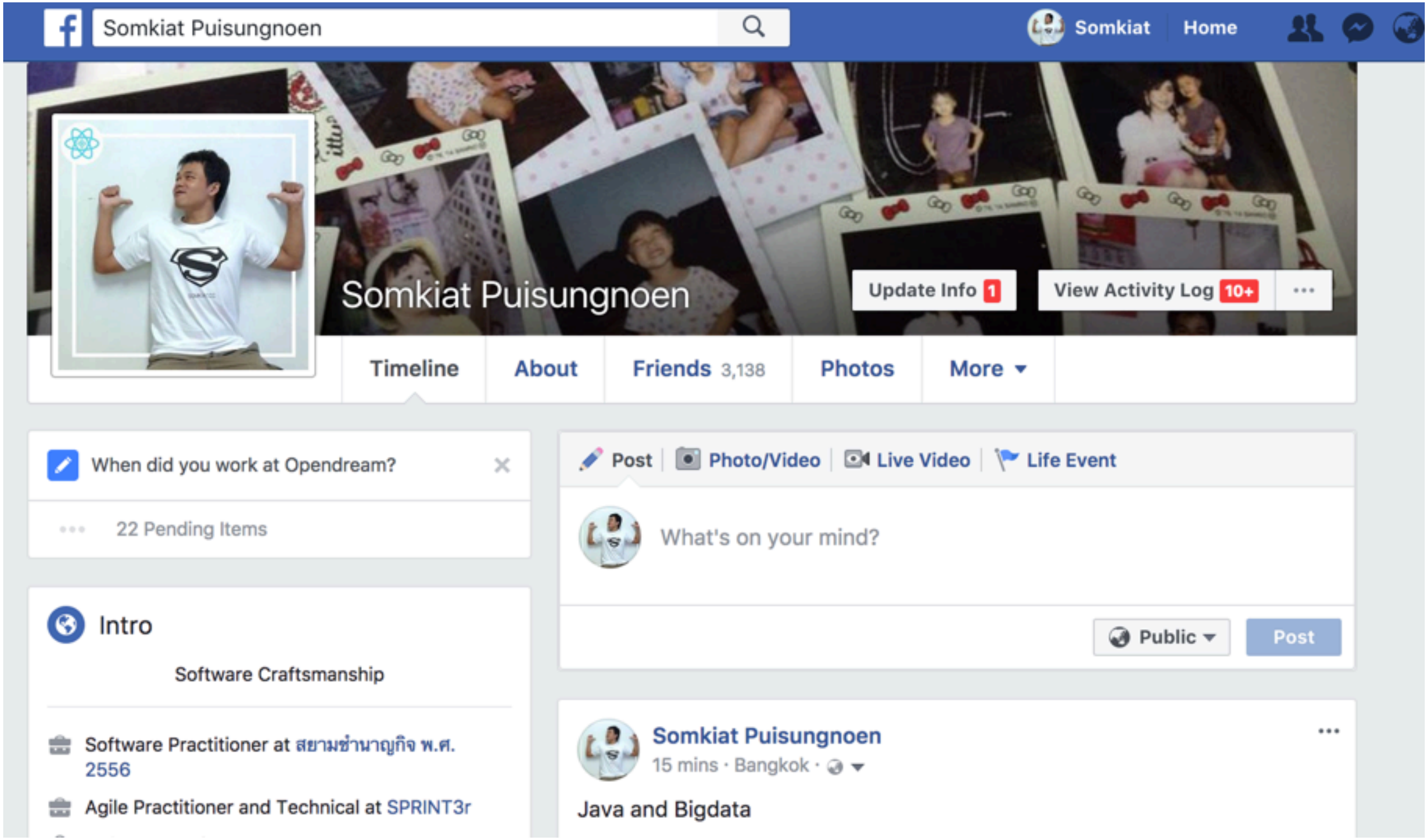
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Solution Architect

สยามชำนาญกิจ
SIAM CHAMNAN KIT

SHU HA RI
SOFTWARE DEVELOPMENT SCHOOL



Somkiat Puisungnoen

When did you work at Opendream?

22 Pending Items

Intro

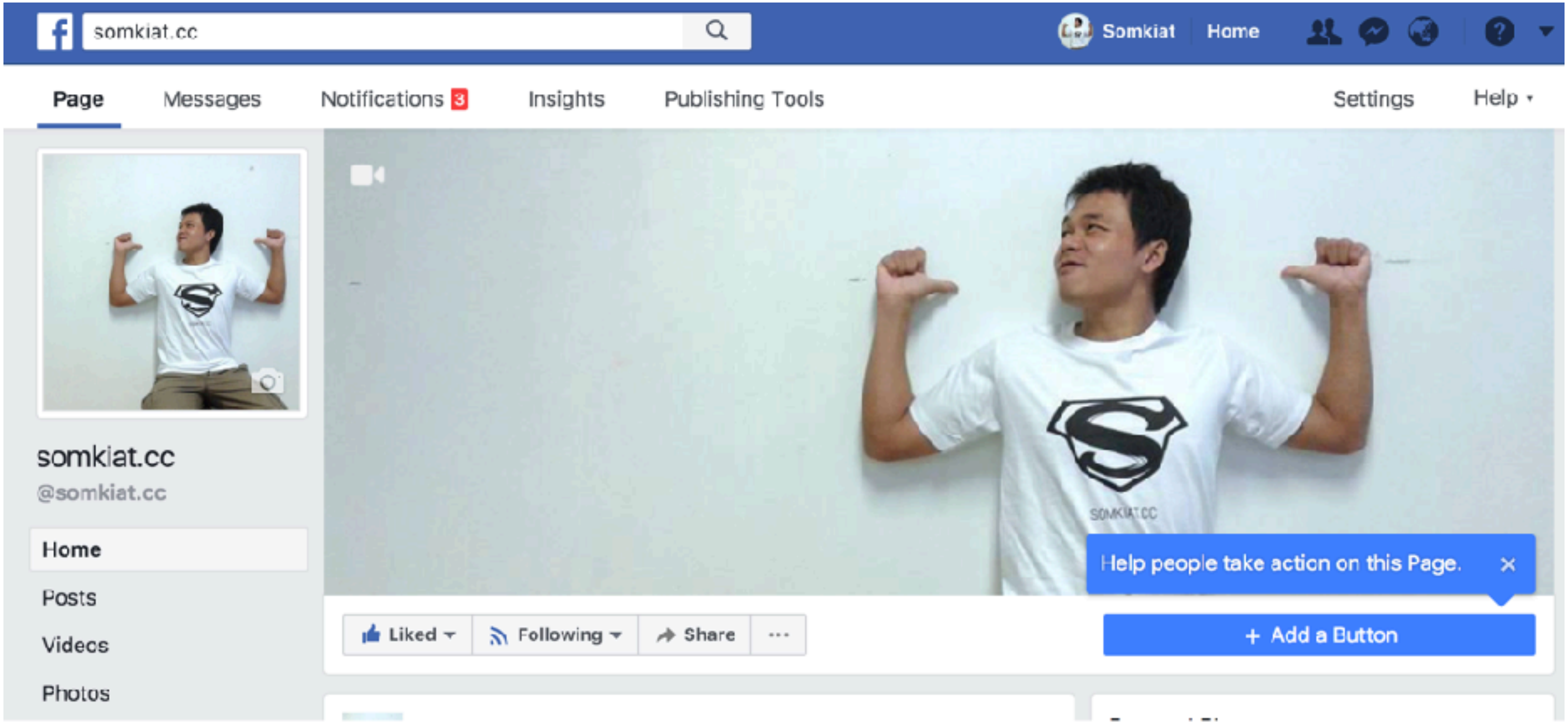
Software Craftsmanship

Software Practitioner at สยามชำนาญกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen
15 mins · Bangkok ·

Java and Bigdata


somkiat.cc
Somkiat
Home
Page
Messages
Notifications 3
Insights
Publishing Tools
Settings
Help
somkiat.cc
@somkiat.cc
Home
Posts
Videos
Photos
Help people take action on this Page.
Liked
Following
Share
+ Add a Button

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

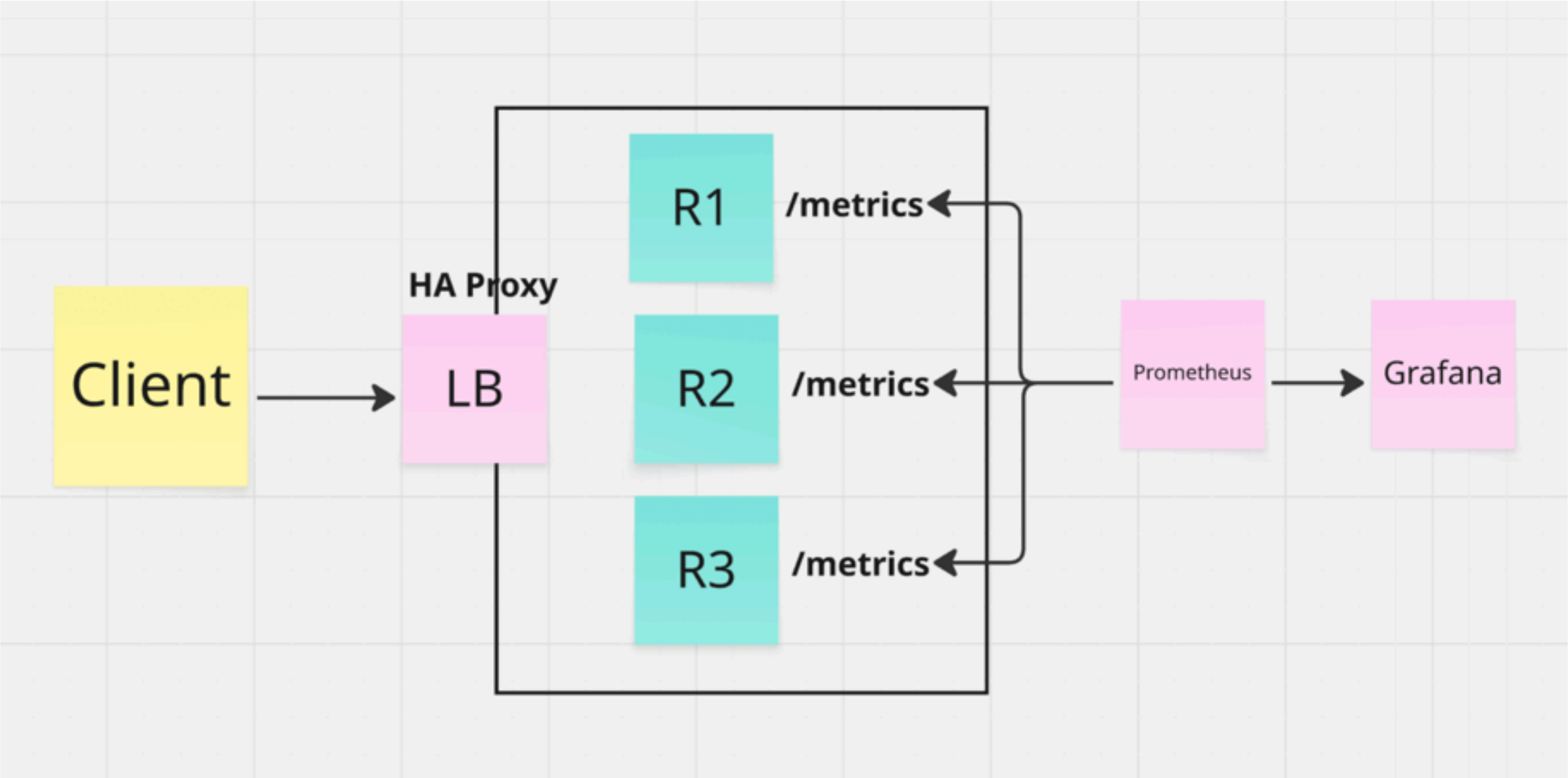
Monitoring and observability

Development

Messaging Queue

Introduction to RabbitMQ

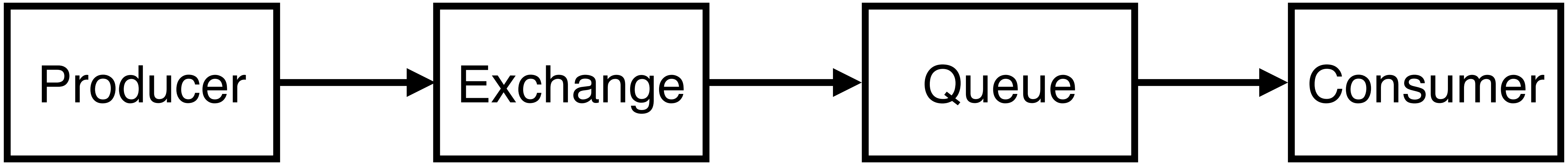
Basic Architecture



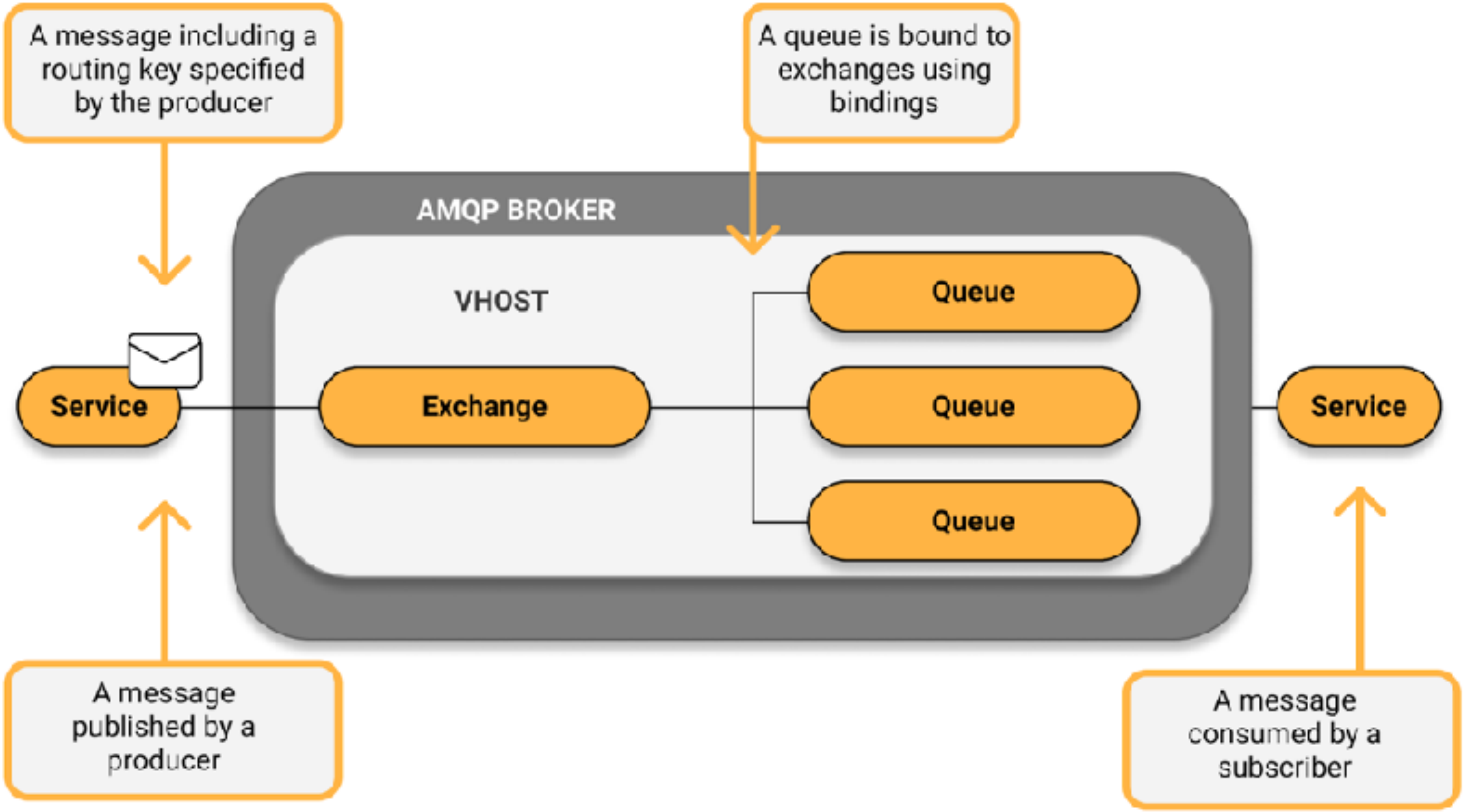
How RabbitMQ works ?

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



RabbitMQ works



A message including a routing key specified by the producer
A queue is bound to exchanges using bindings
AMQP BROKER
VHOST
Service
Exchange
Queue
Queue
Queue
Service
A message published by a producer
A message consumed by a subscriber

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 parameters

Parameter	Description	Use case
durable	**true** = queue will survive a broker restart	Persistent work queue where don’t want data loss
exclusive	**true** = queue is restricted to the connection that declare it will delete when connection loss	One queue per connection
auto-delete	**true** = queue is automatically deleted when consumer unsubscribed	Temporary notification queue

Queue arguments (optional)

Name	Description	Use case
x-queue-mode	**default = in-memory** lazy = pages to disk	Very large queue Memory pressure Long running queue
x-queue-expire	Time(in ms) for queue may be unused Before auto-deleted	Cleanup unused queue
x-max-length	Maximum number of messages in queue	Keep only the latest N datas
x-message-ttl	Time(in ms) for message live in queue, before discarded or dead-lettered	Message expiration Retry message
x-max-priority	Enable prioritized queue (0 to N)	High priority jobs Critical alert

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

https://www.rabbitmq.com/docs/quorum-queues

Quorum queue

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

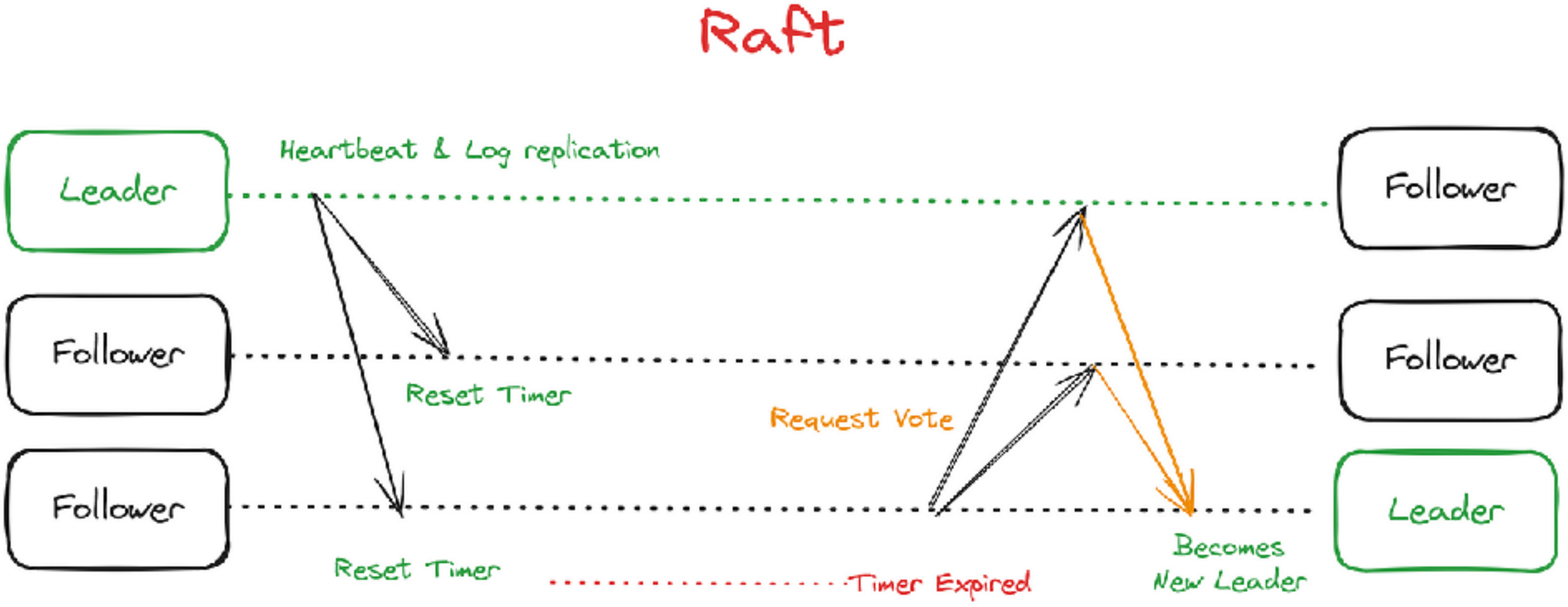
High latency !!

Data loss is unacceptable
(Financial data and sensitive data)

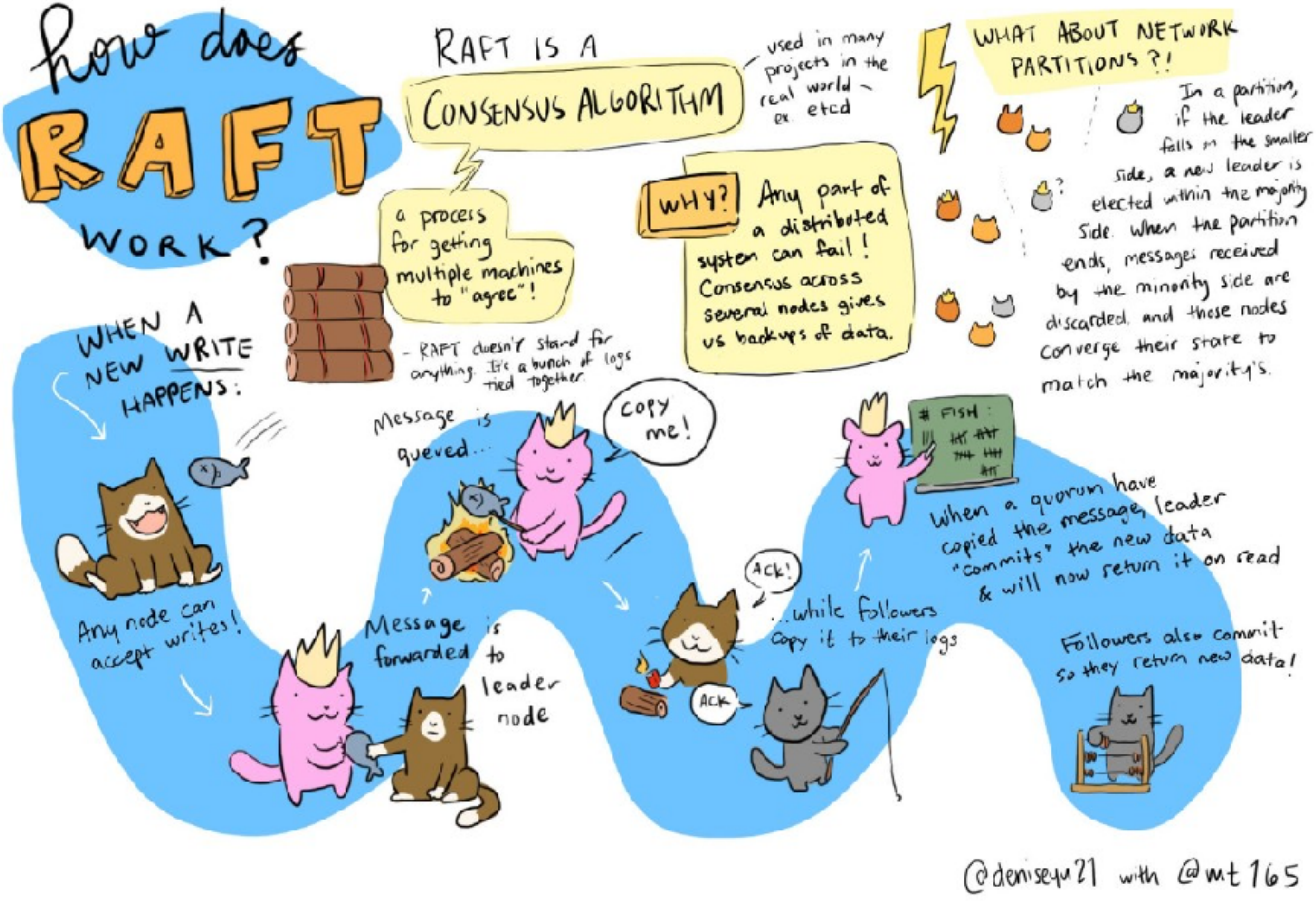
(N/2)+1

https://www.rabbitmq.com/docs/quorum-queues

Raft consensus ?



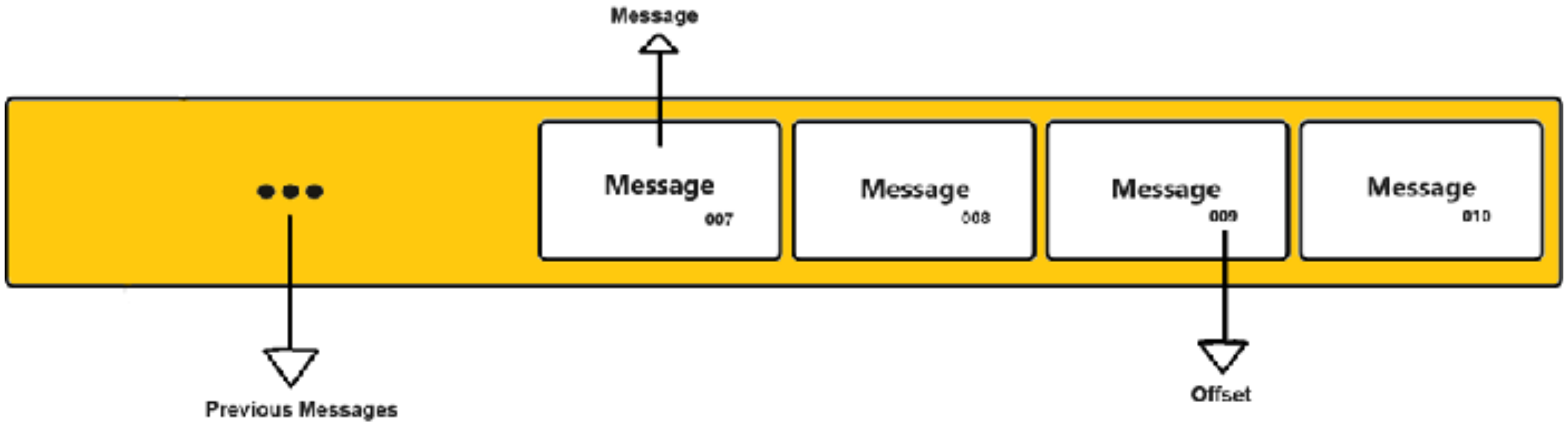
https://en.wikipedia.org/wiki/Raft_(algorithm)


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"!
- RAFT doesn't stand for anything. It's a bunch of logs tied together.
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 on 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 is forwarded to leader node
Message is queued...
Copy me!
Ack!
...while followers copy it to their logs
ACK
# FISH
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!
@deniseyu21 with @mt165

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

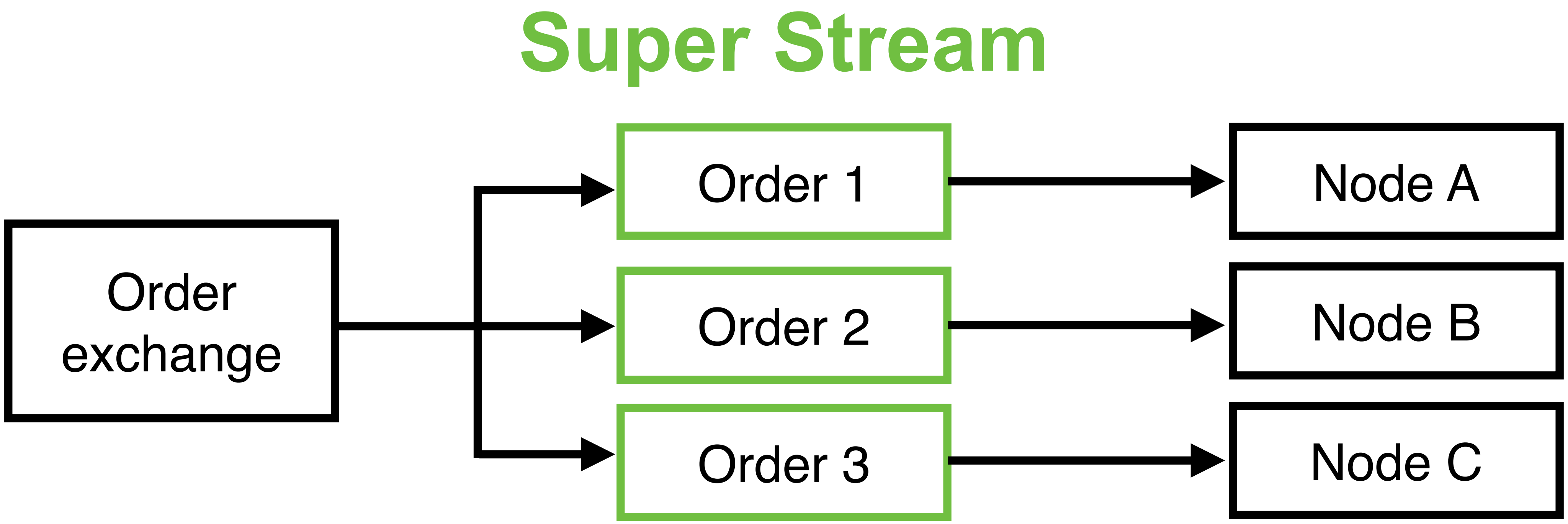
Stream

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

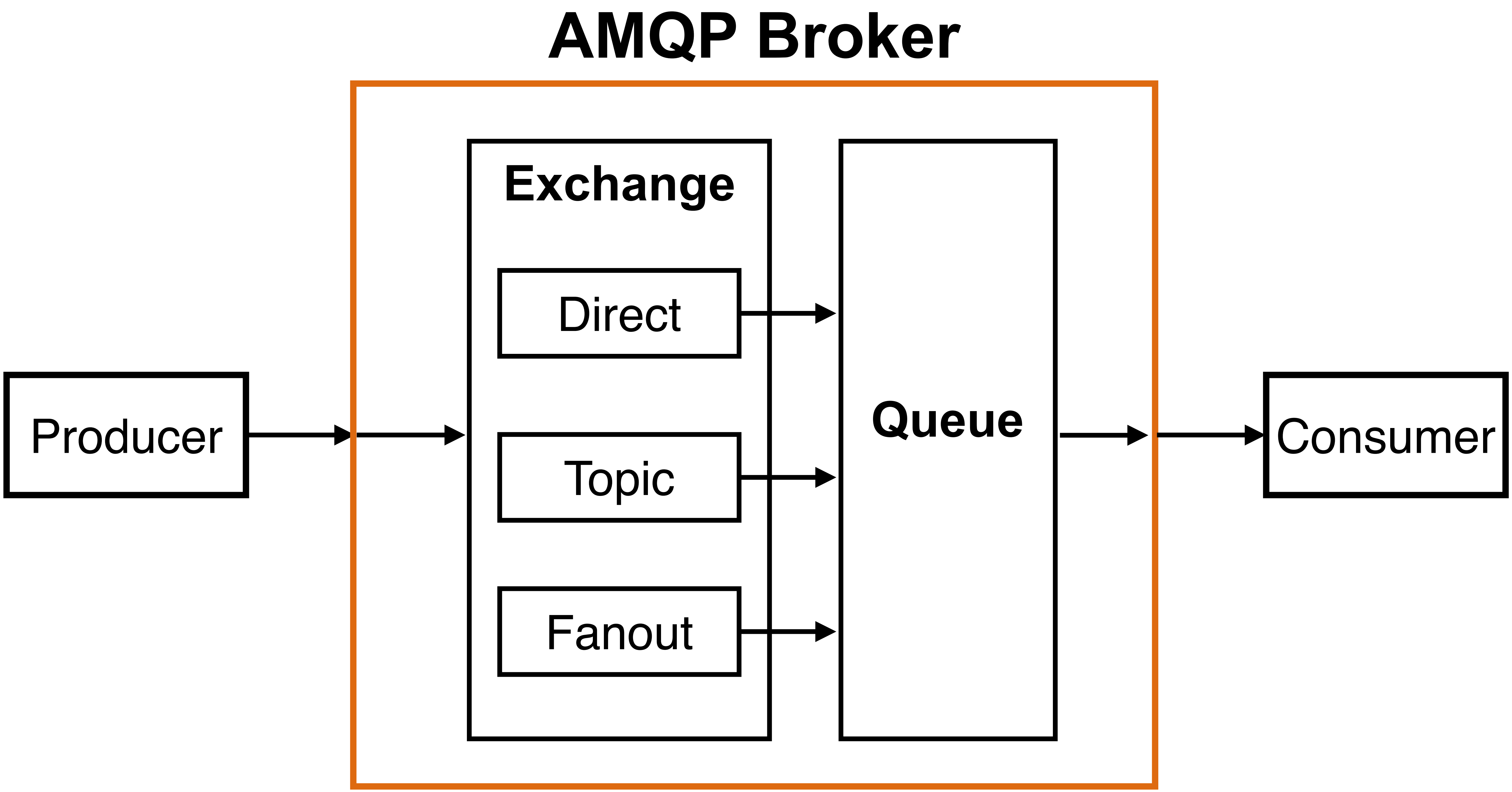


https://www.rabbitmq.com/blog/2022/07/13/rabbitmq-3-11-feature-preview-super-streams

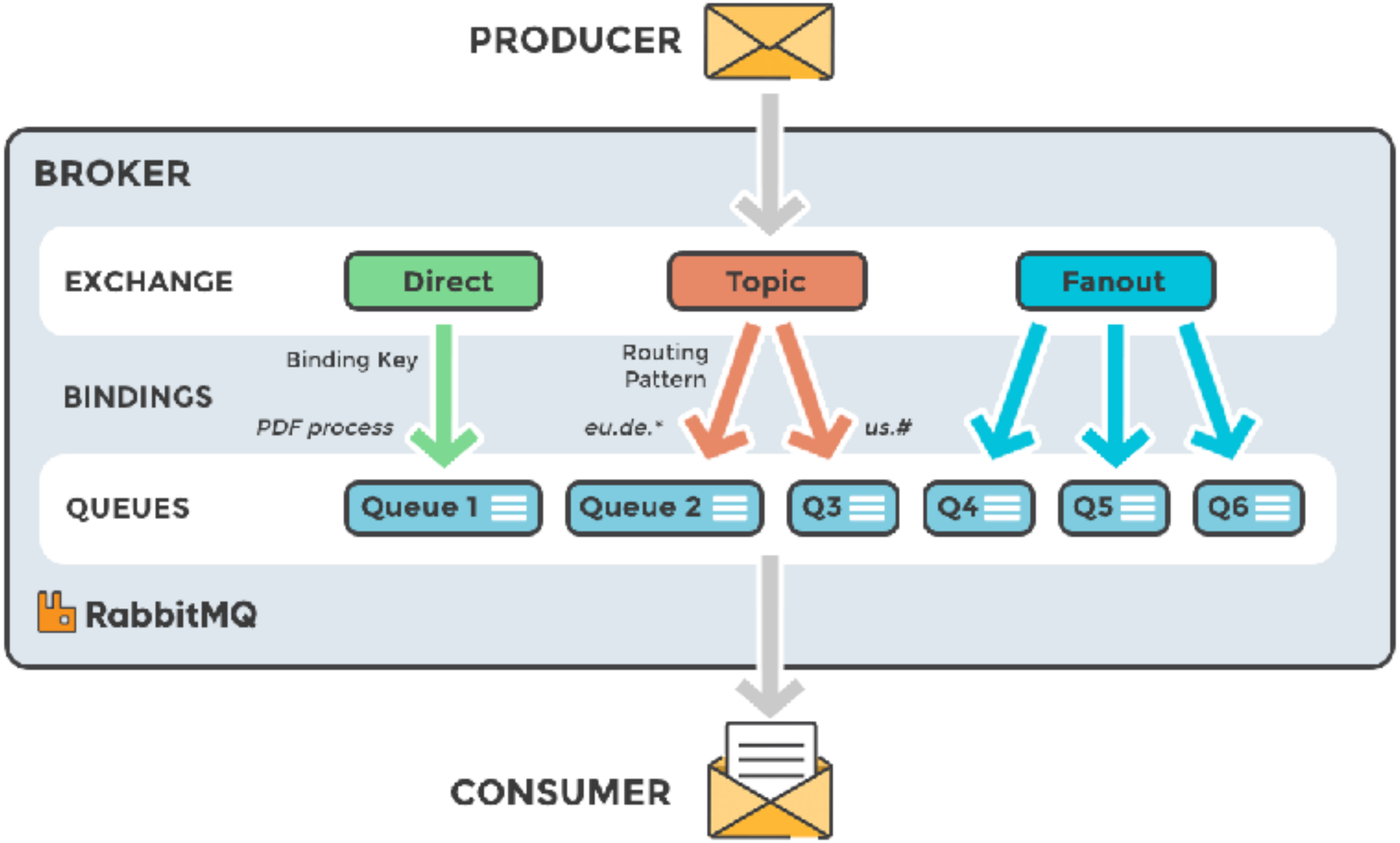
Working with Classic queue

RabbitMQ works

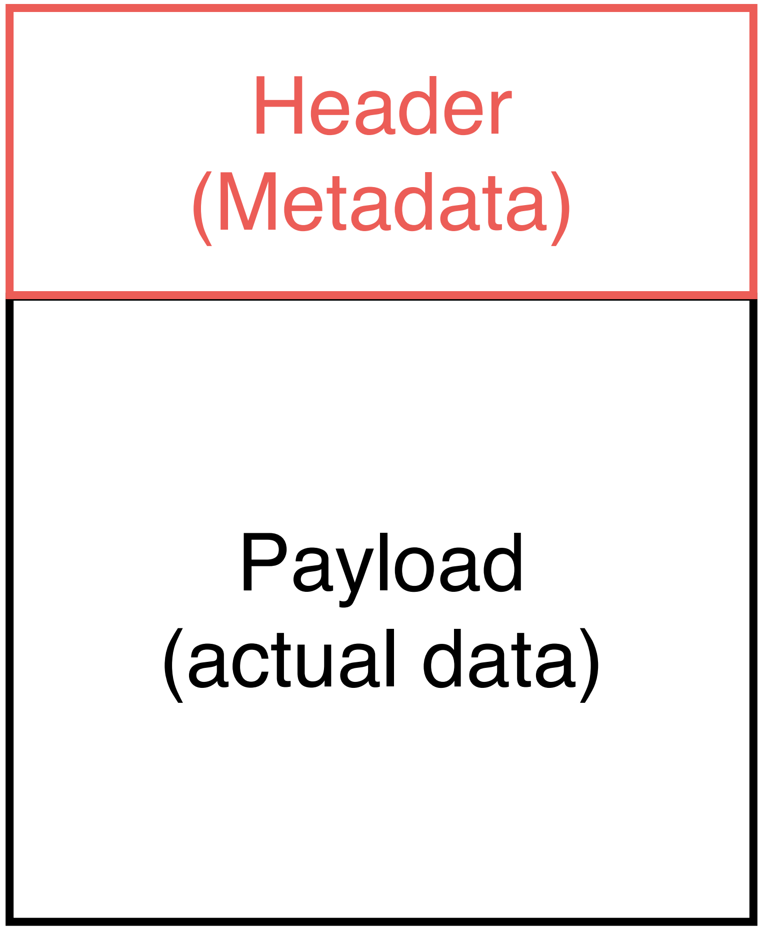
RabbitMQ works



RabbitMQ works



Message structure

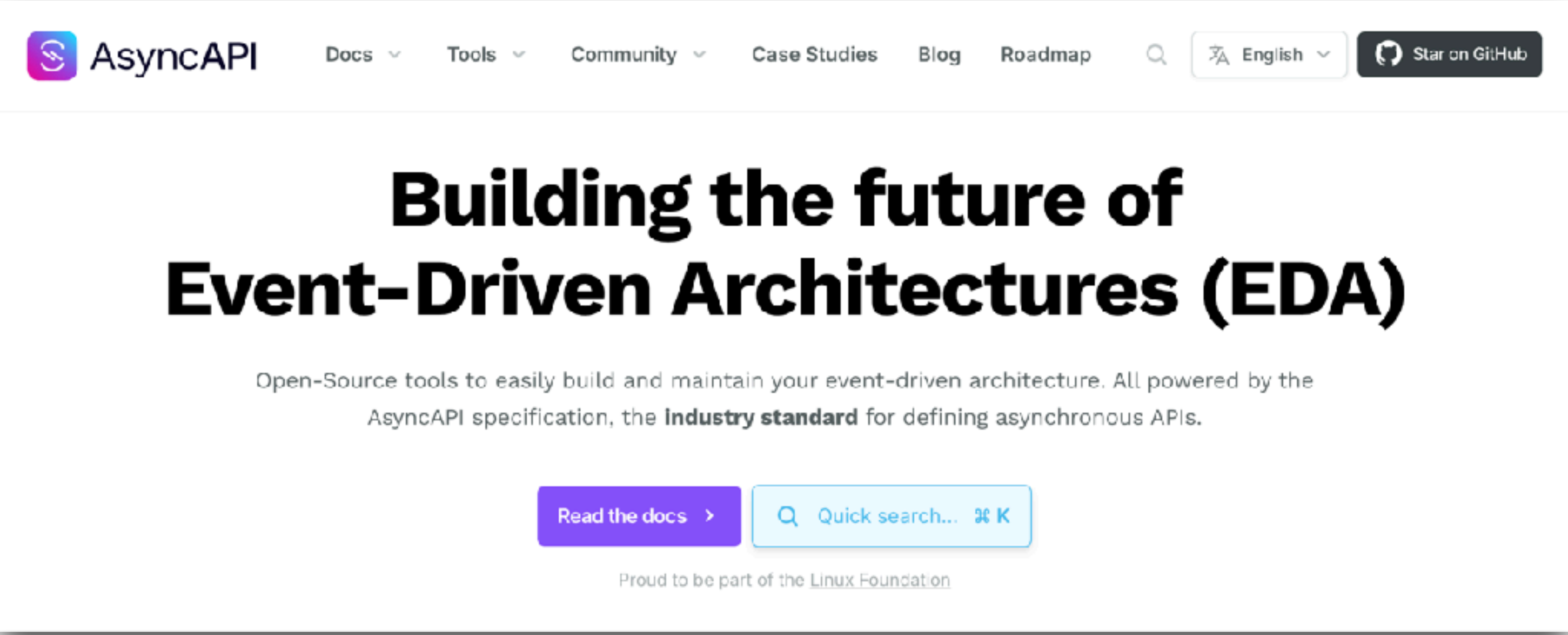


message id
content type
priority
routing key

https://www.rabbitmq.com/docs/consumers#message-properties

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.

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

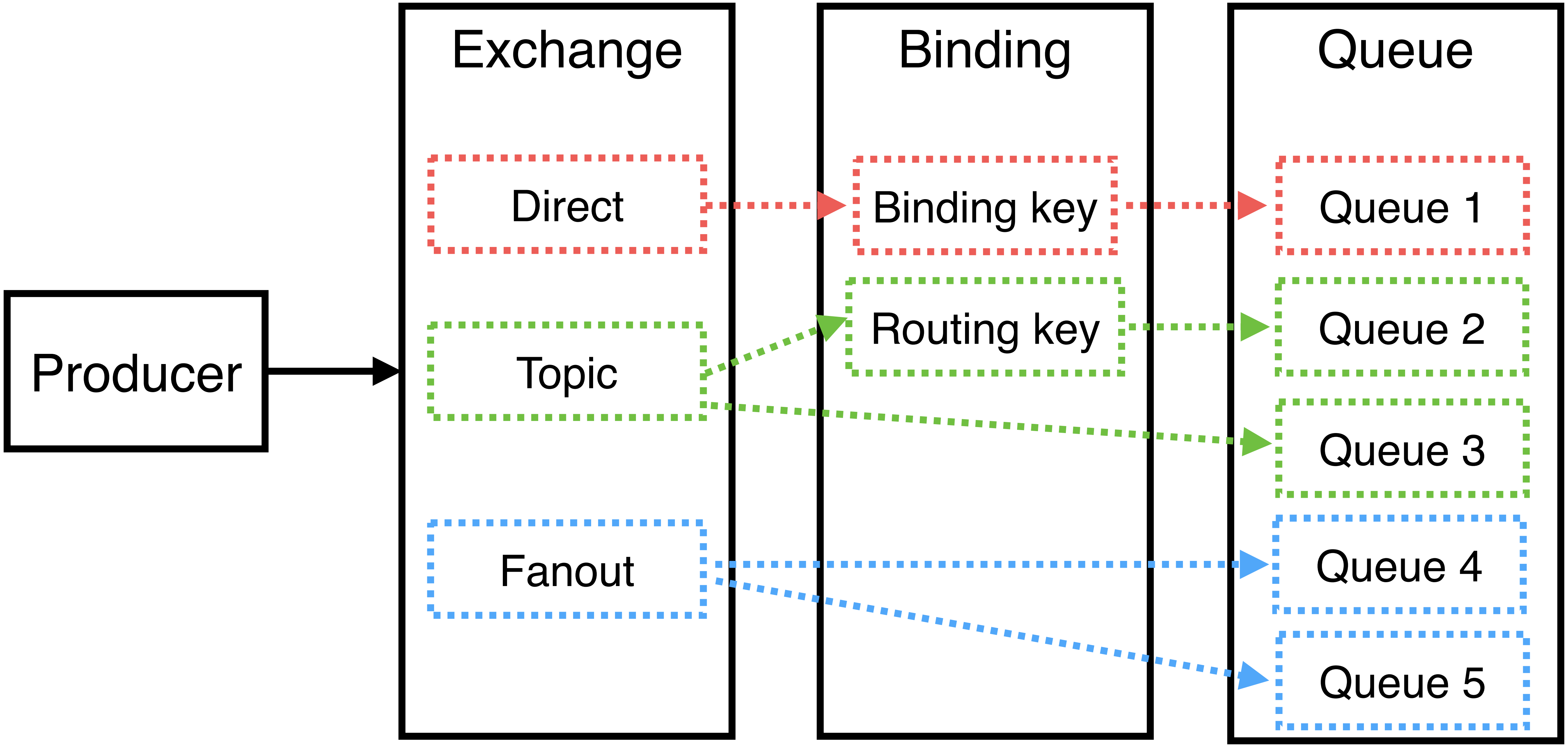
Direct

Fanout

Topic

Header

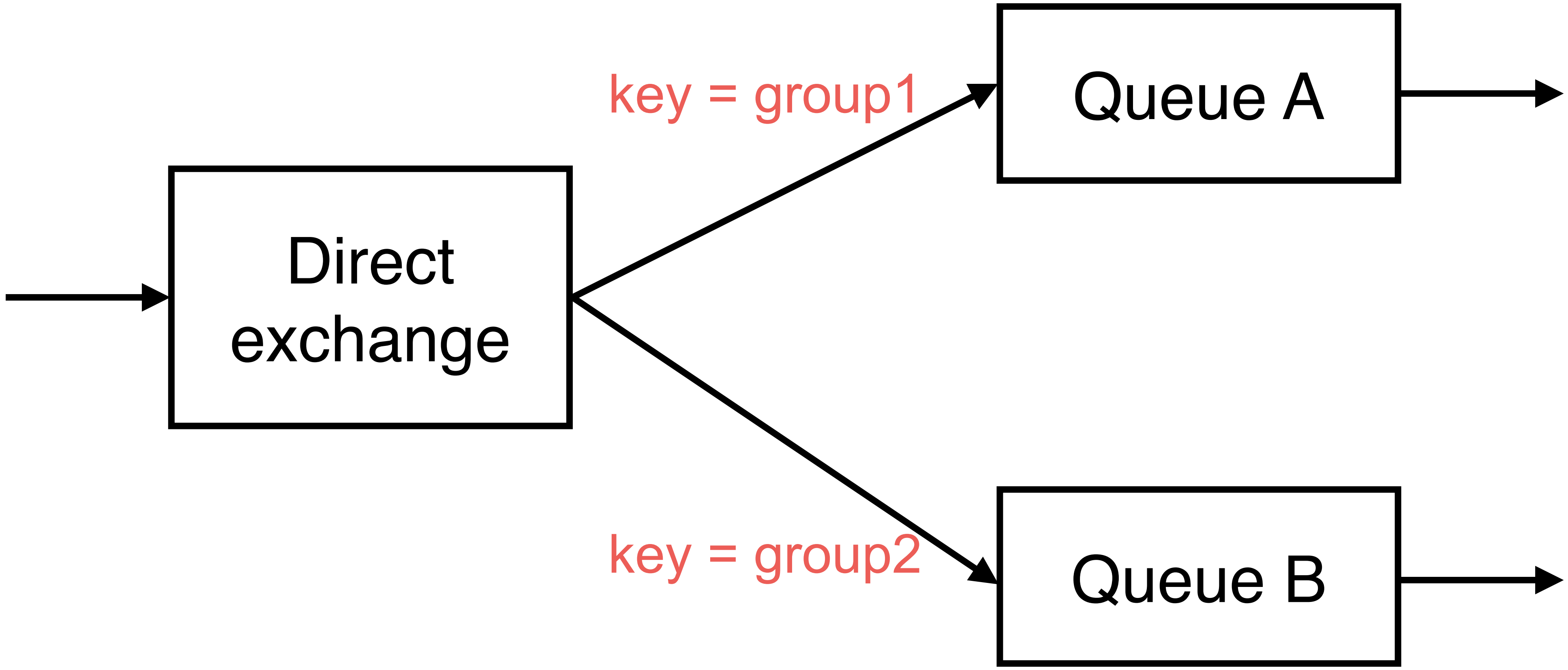
Exchange ?


Exchange
Binding
Queue
Direct
Binding key
Queue 1
Producer
Topic
Routing key
Queue 2
Queue 3
Fanout
Queue 4
Queue 5

Direct Exchange

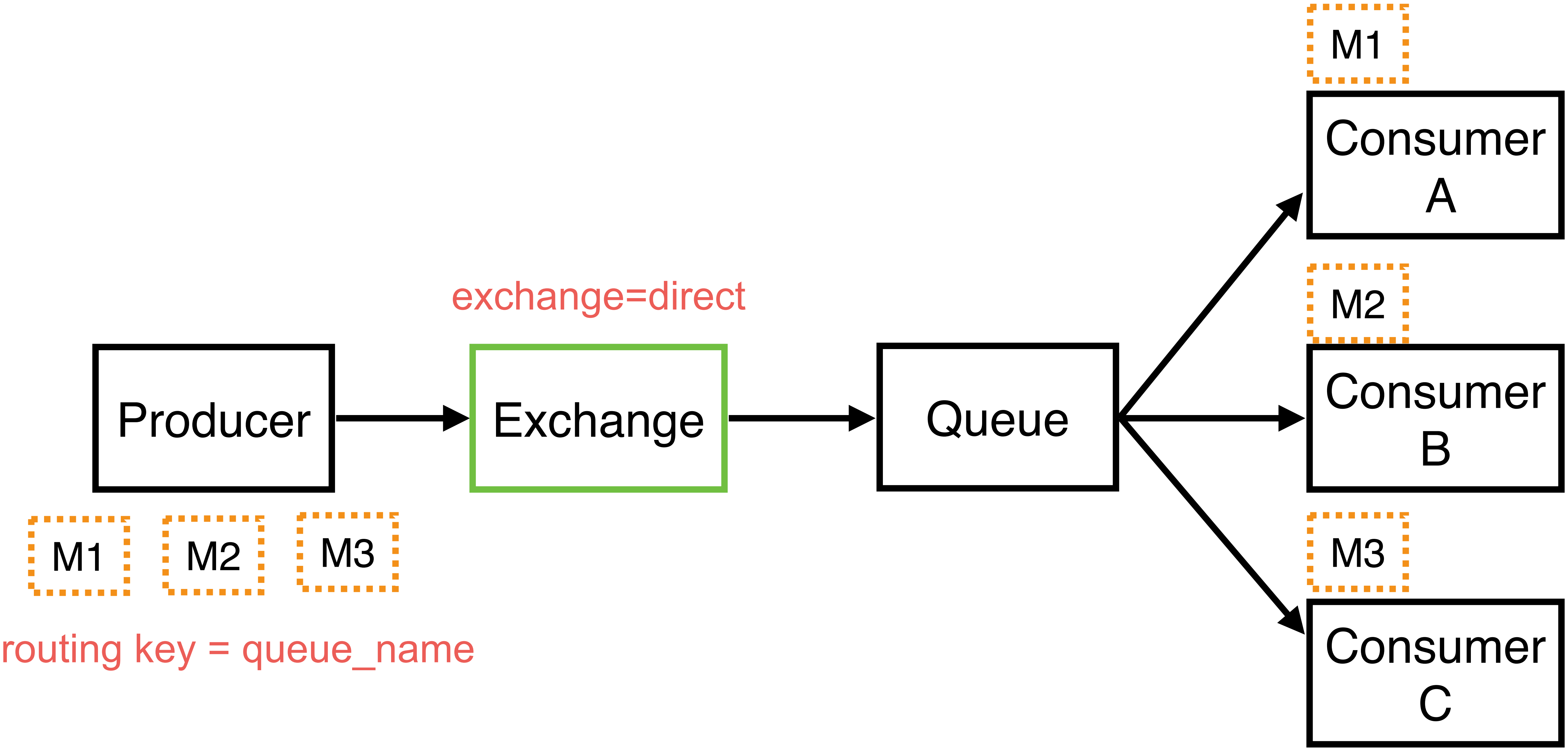
Direct exchange

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



1. Work Queues

Exchange = empty = direct



exchange=direct

Producer

Exchange

Queue

Consumer A

Consumer B

Consumer C

M1 M2 M3

M1

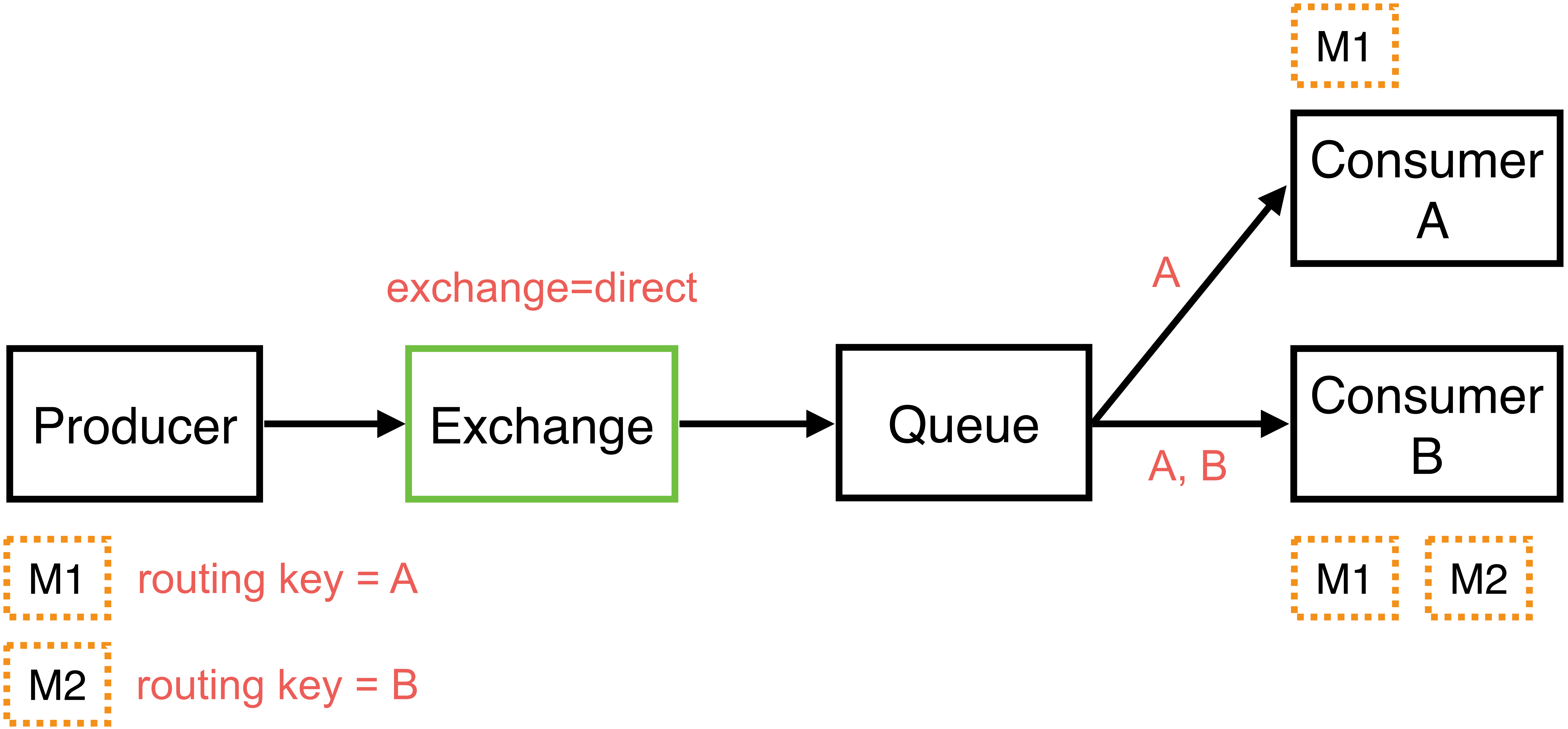
M2

M3

routing key = queue_name

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

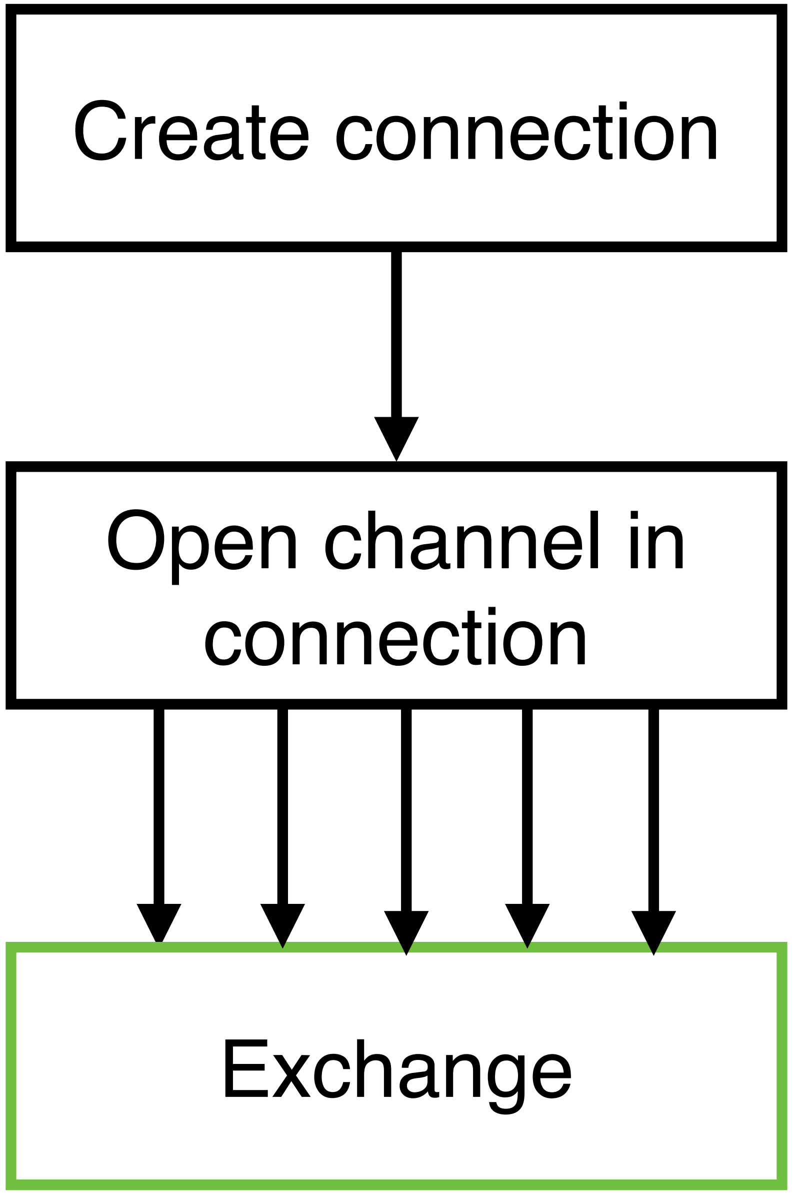
2. Routing by key



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

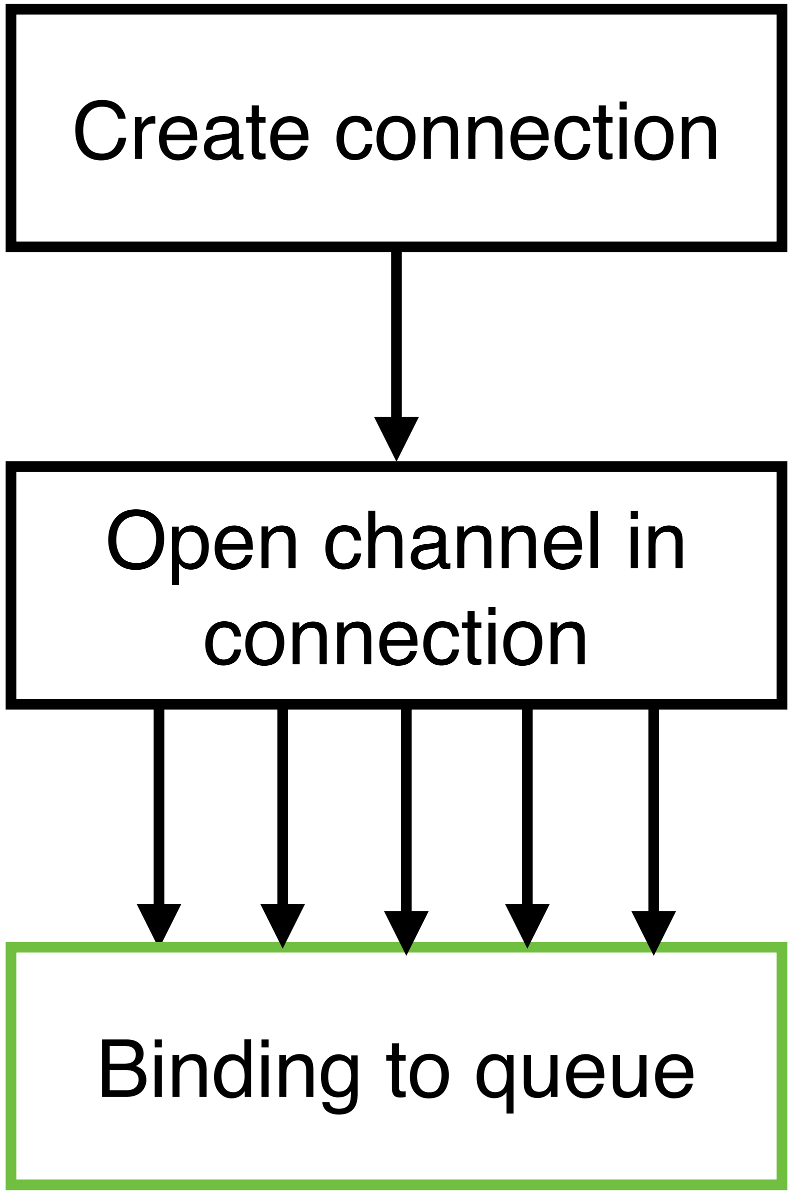
Producer

Step to publish message to exchange



Consumer

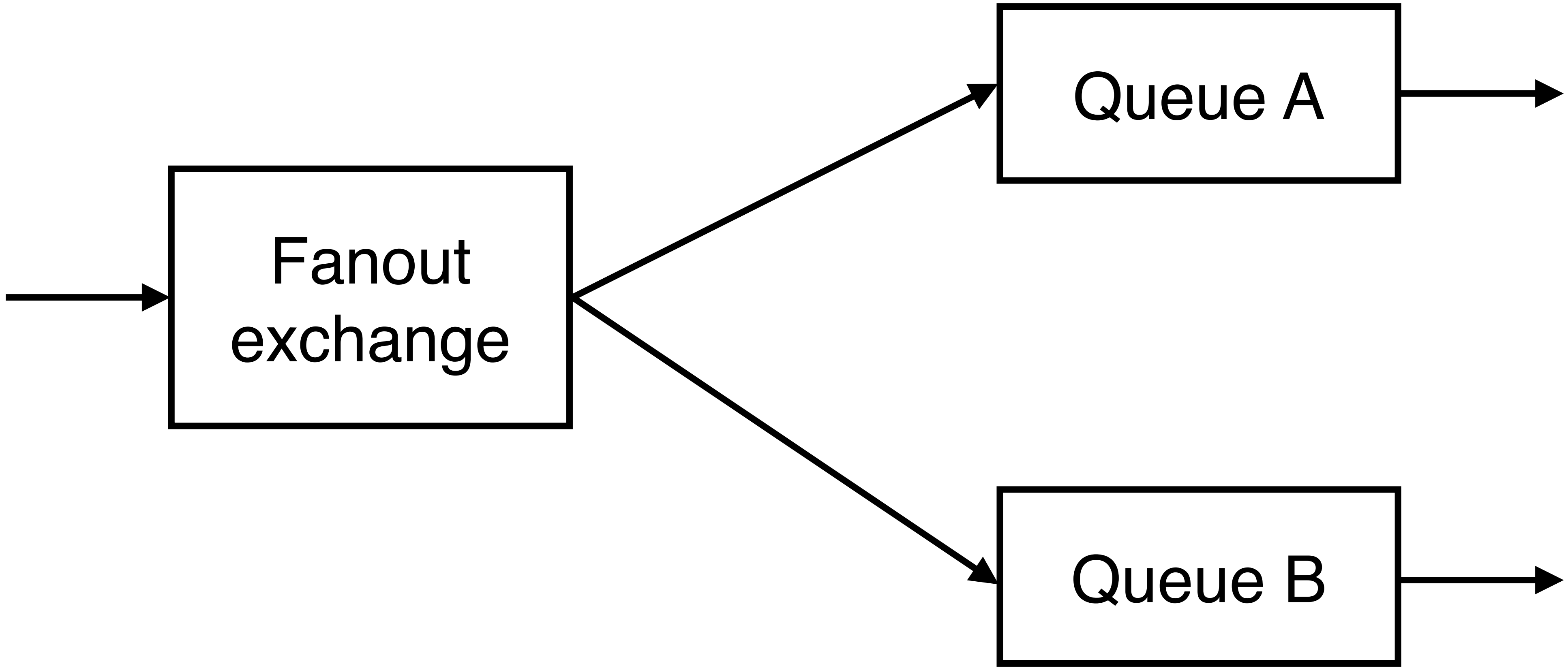
Step to read message from queue



Fanout Exchange

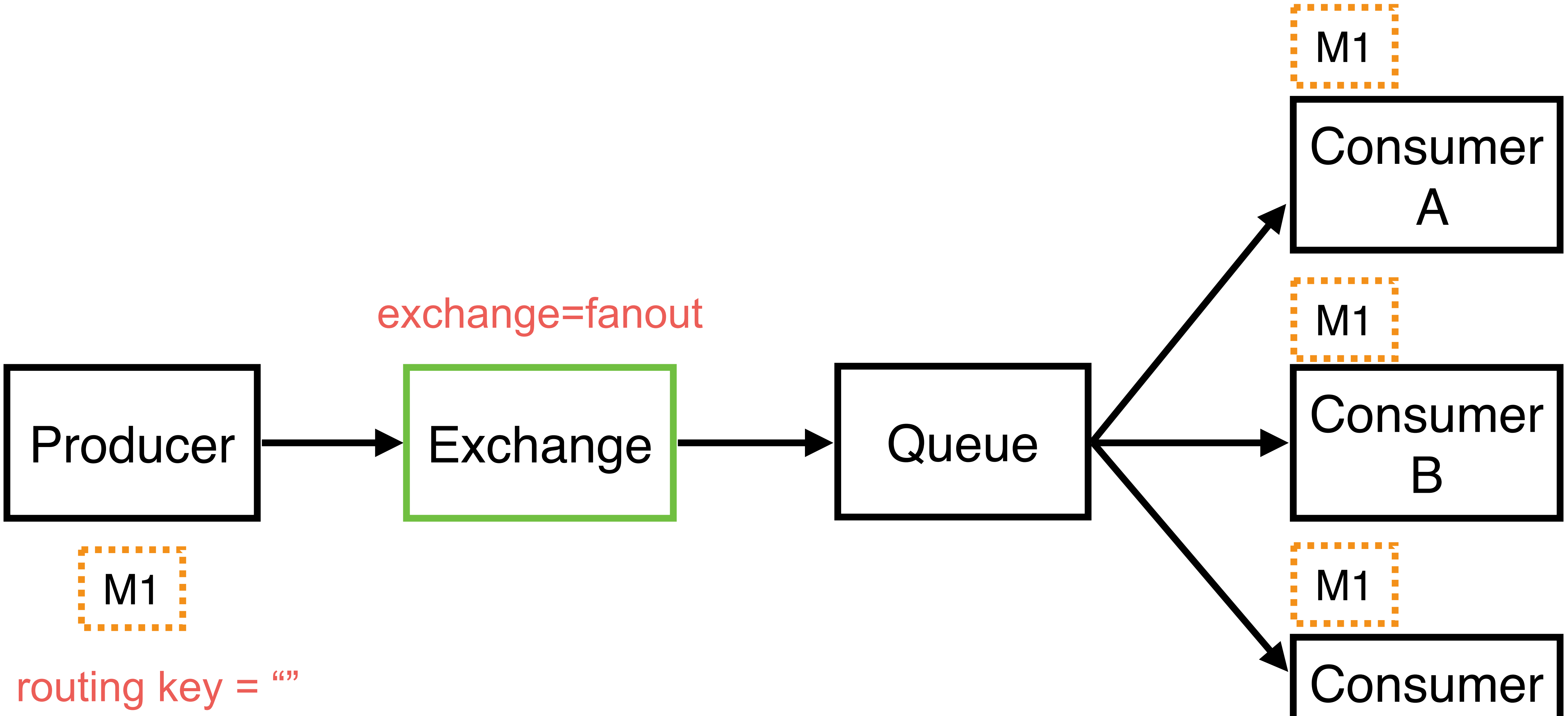
Fanout exchange

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



Publish/Subscribe

Exchange = fanout



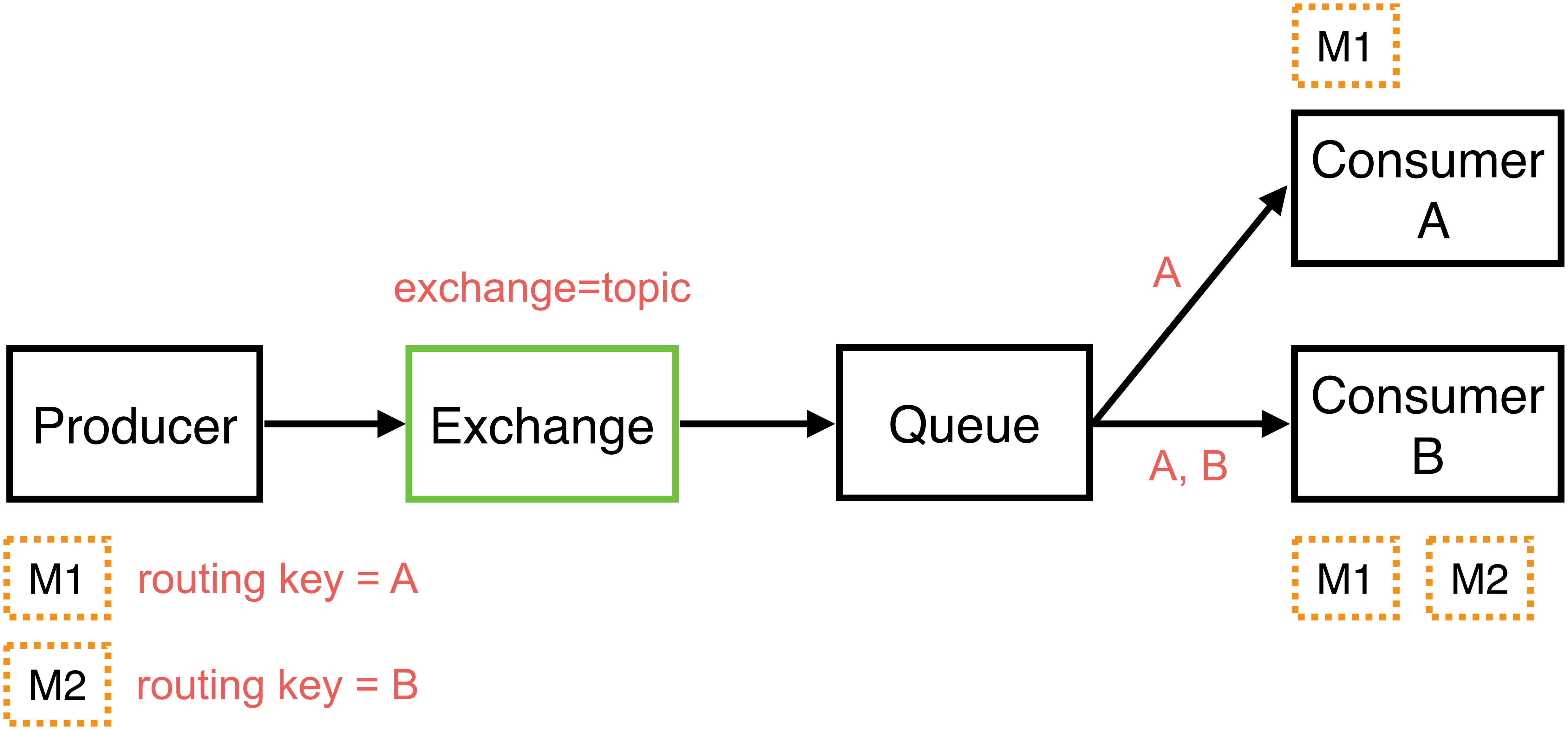
https://github.com/up1/course-rabbitmq-2025/tree/main/workshop/demo-go/pubsub

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

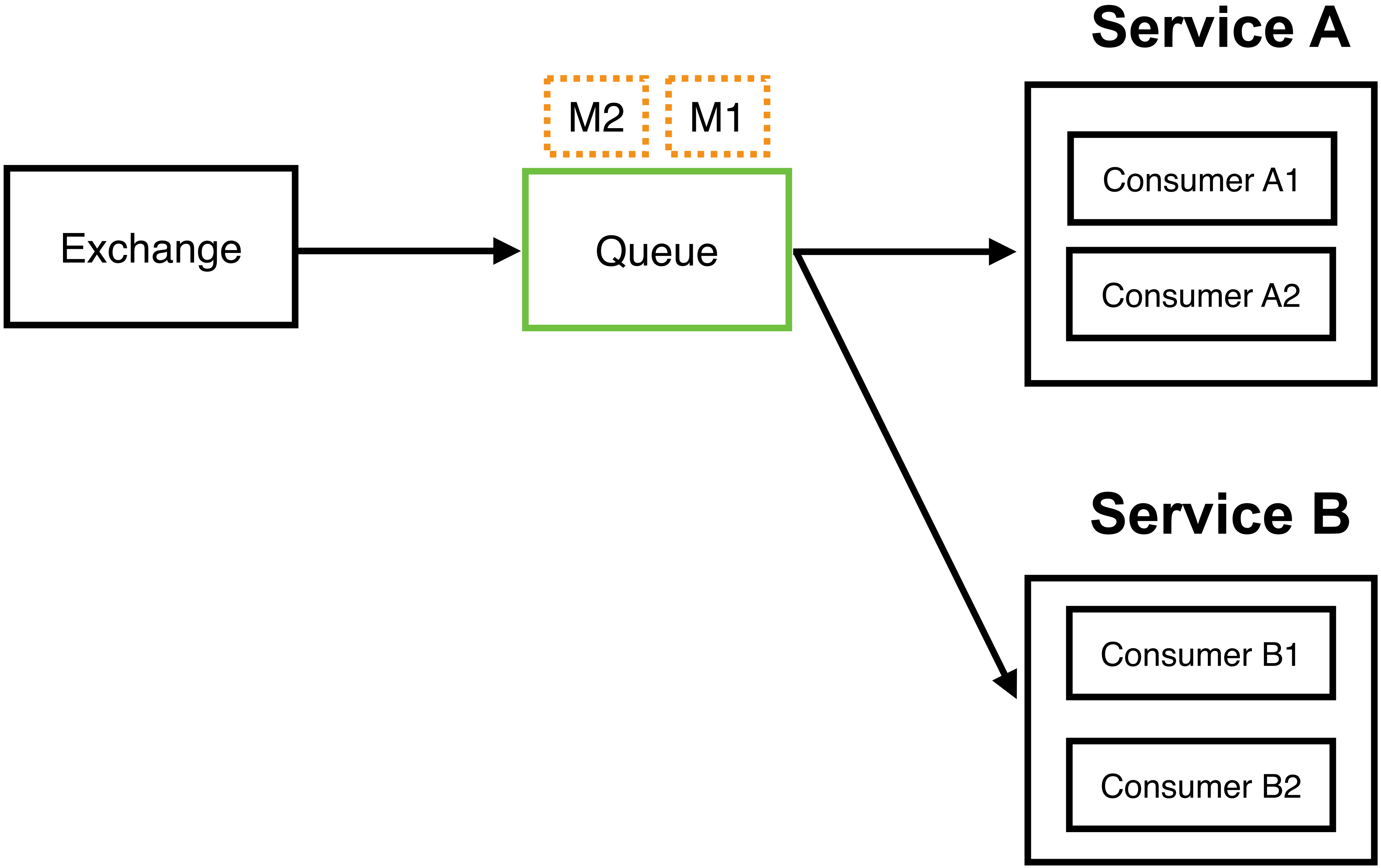
Header Exchange

Header exchange

Using message’s header to routing
Criteria based on header

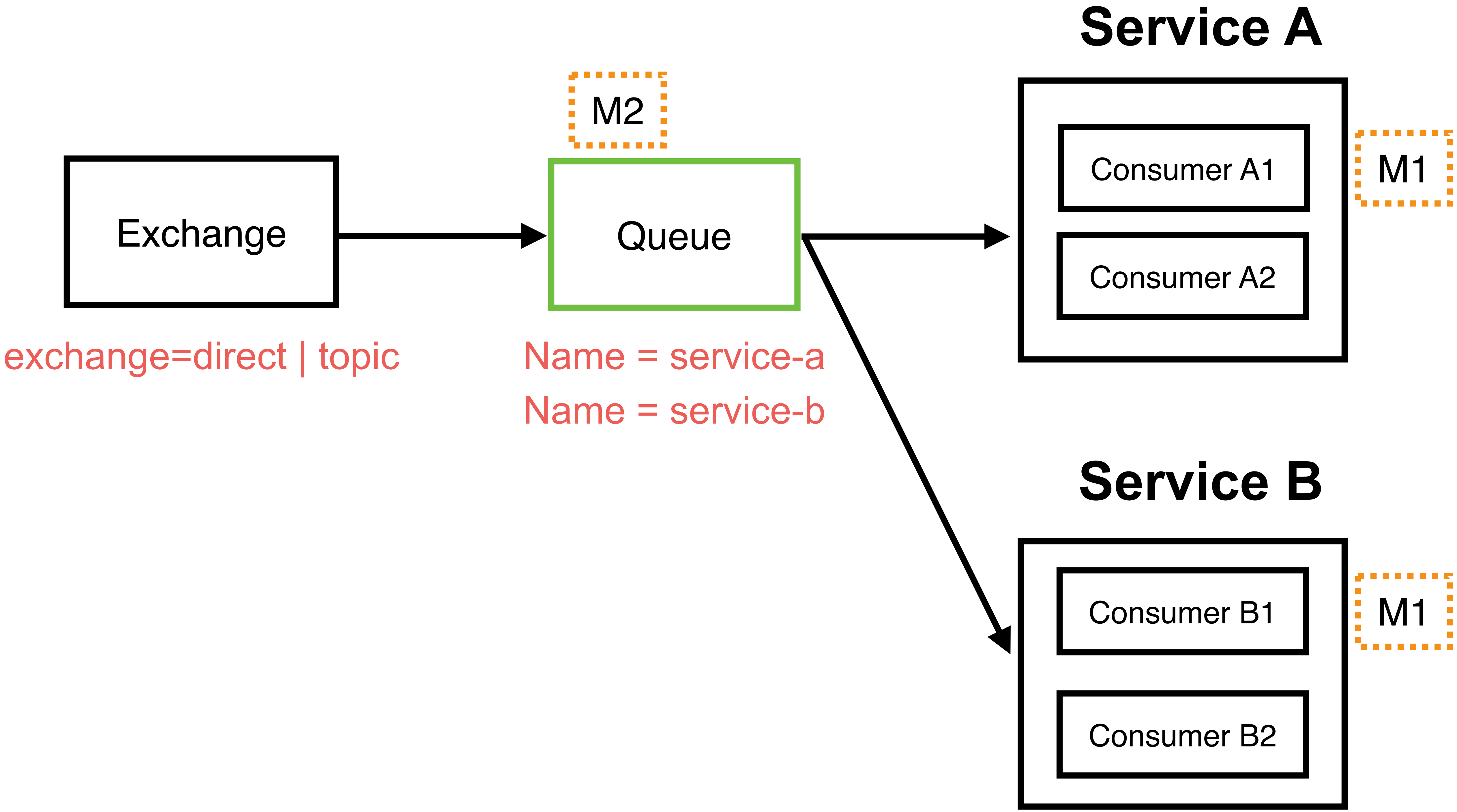
Use Cases

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)

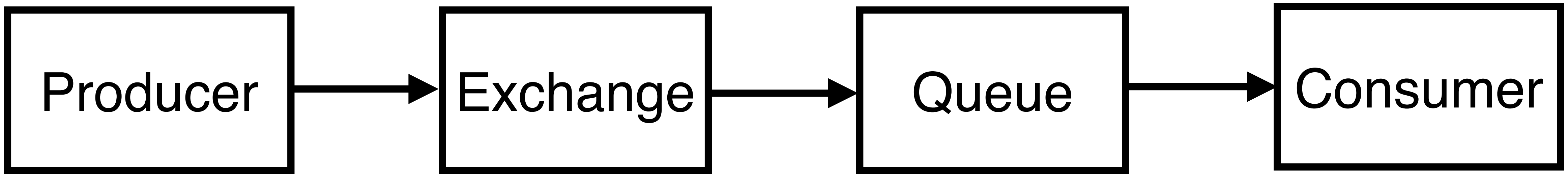
Broker restart/crash

Retry with delay (N time)

Exceeding retry attempts

High-priority Dead Letter Queue (DLQ)

Consumer crash (re-deliver)



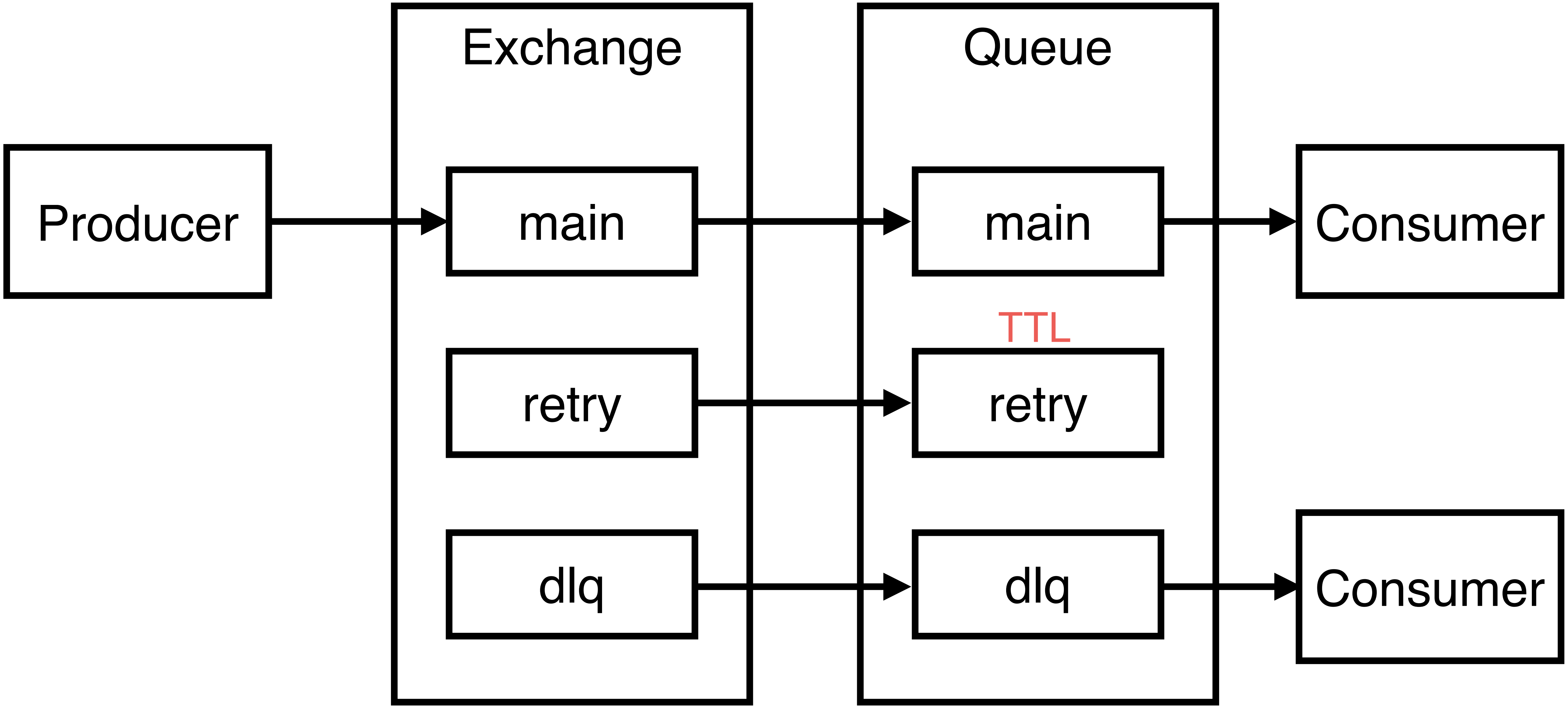
Delivery mode = Persistent

Durable = true
Delete = false

Auto ack = false

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

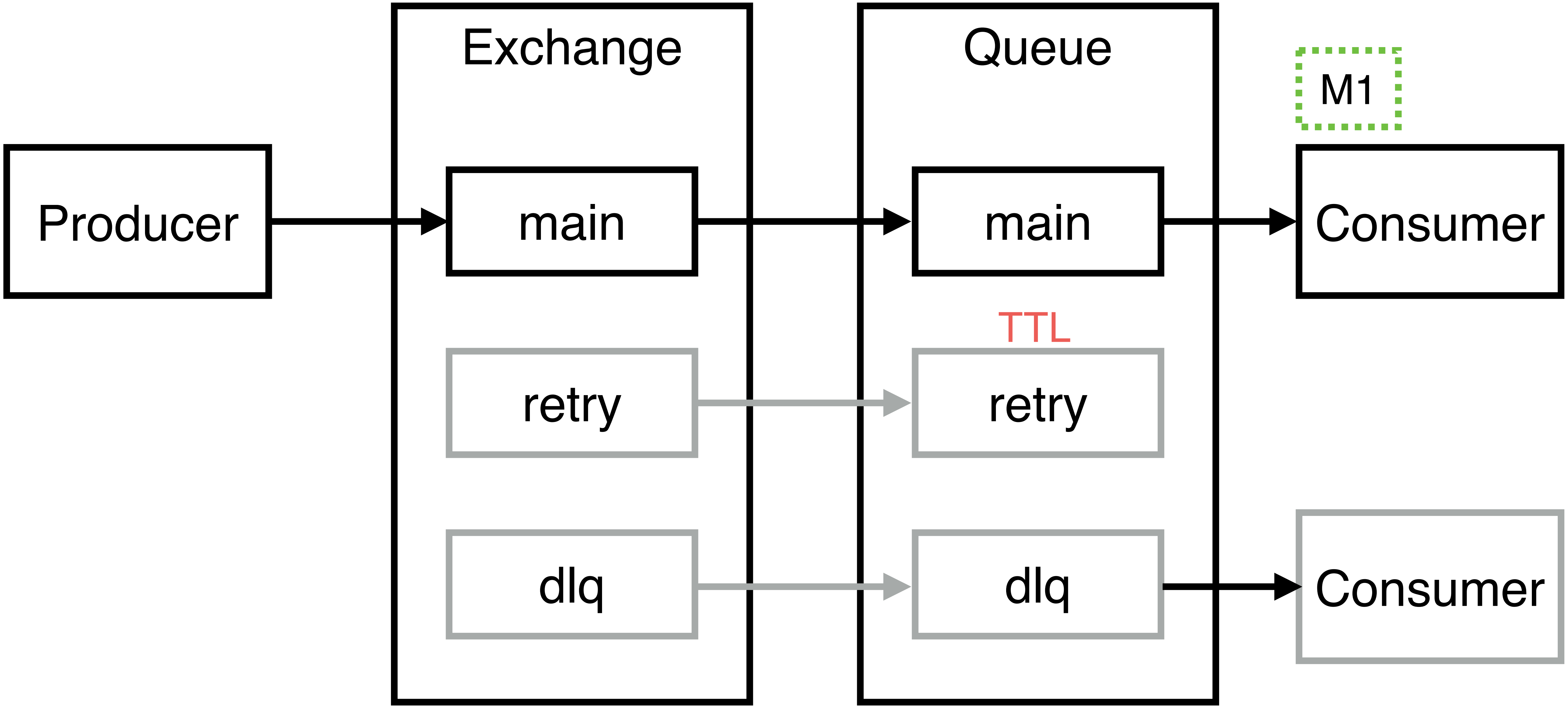
Retry and Dead Letter Queue



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

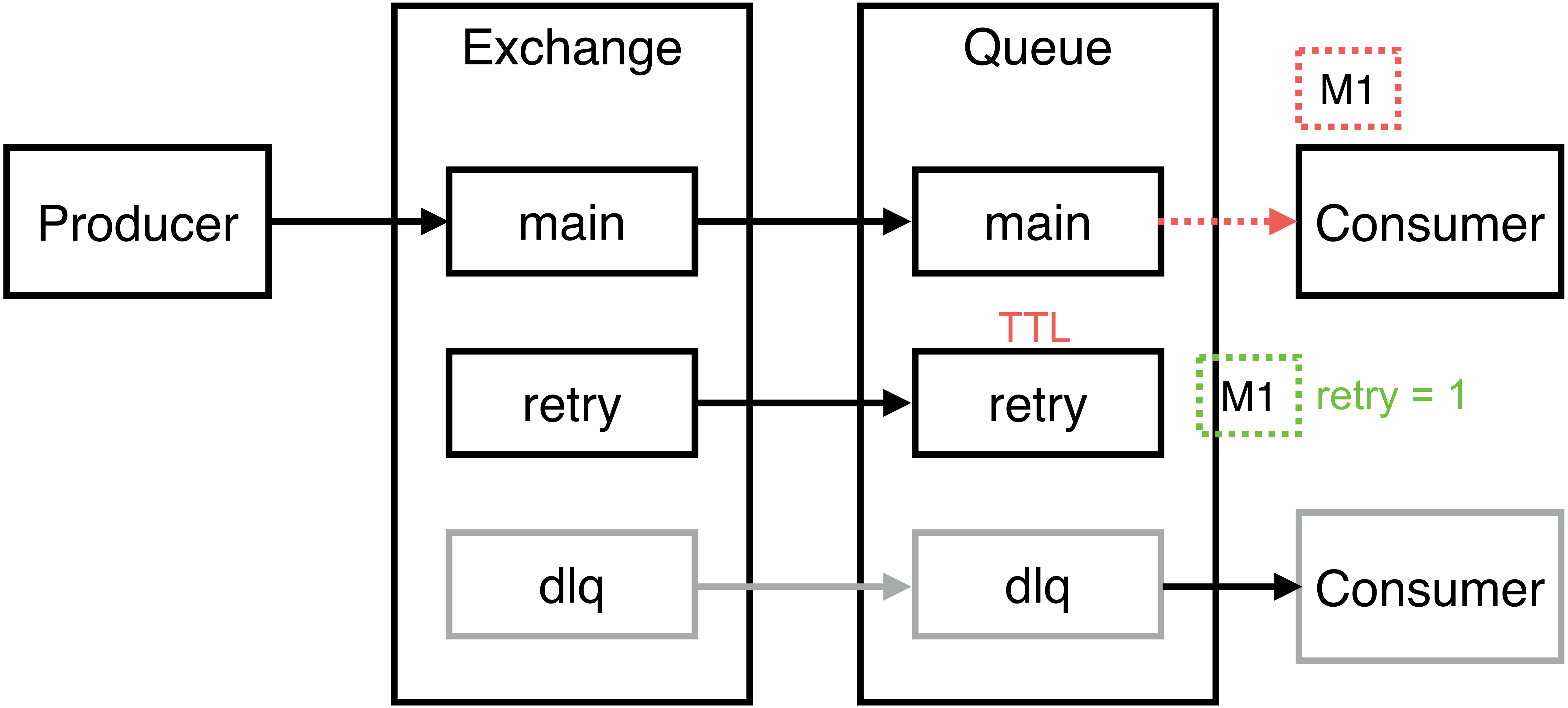
Retry and Dead Letter Queue

Normal process



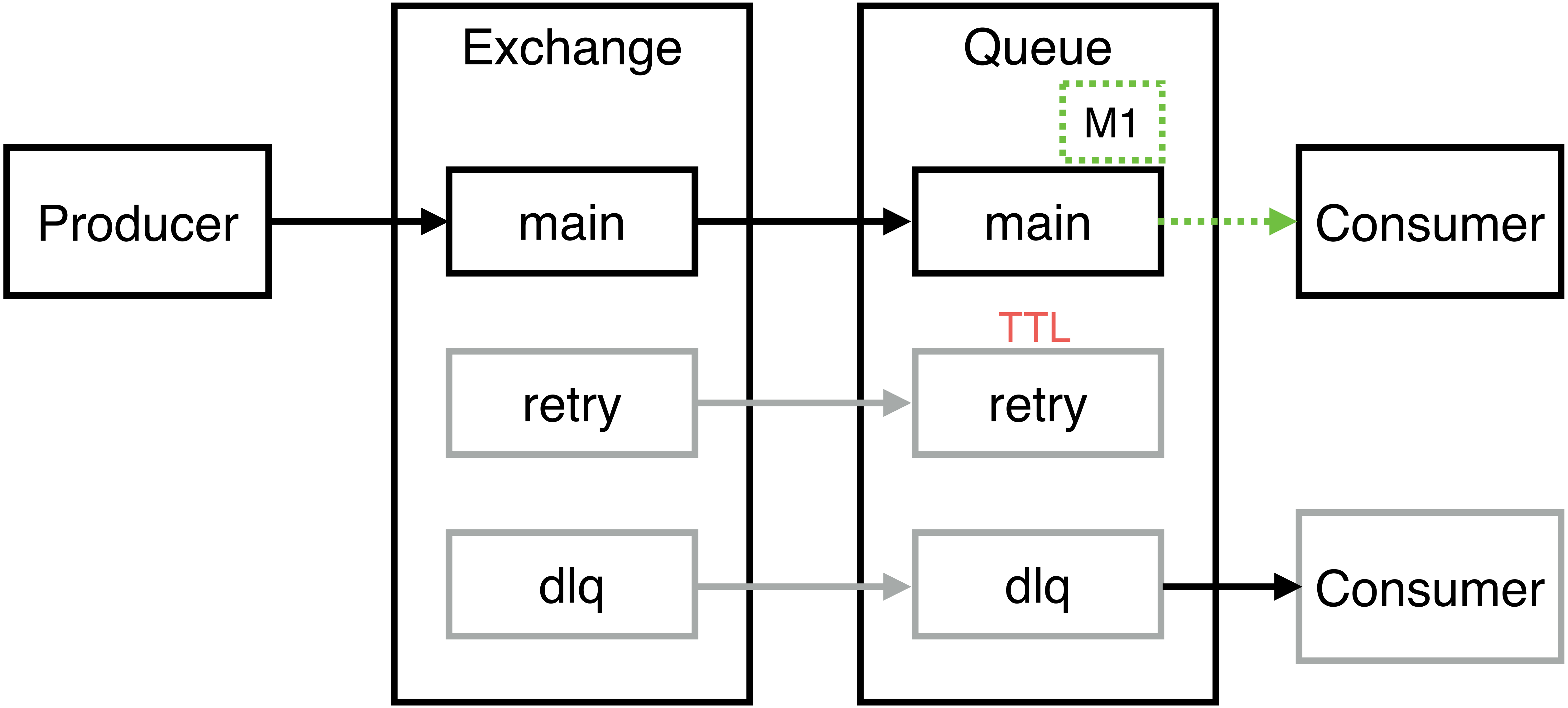
Retry and Dead Letter Queue

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



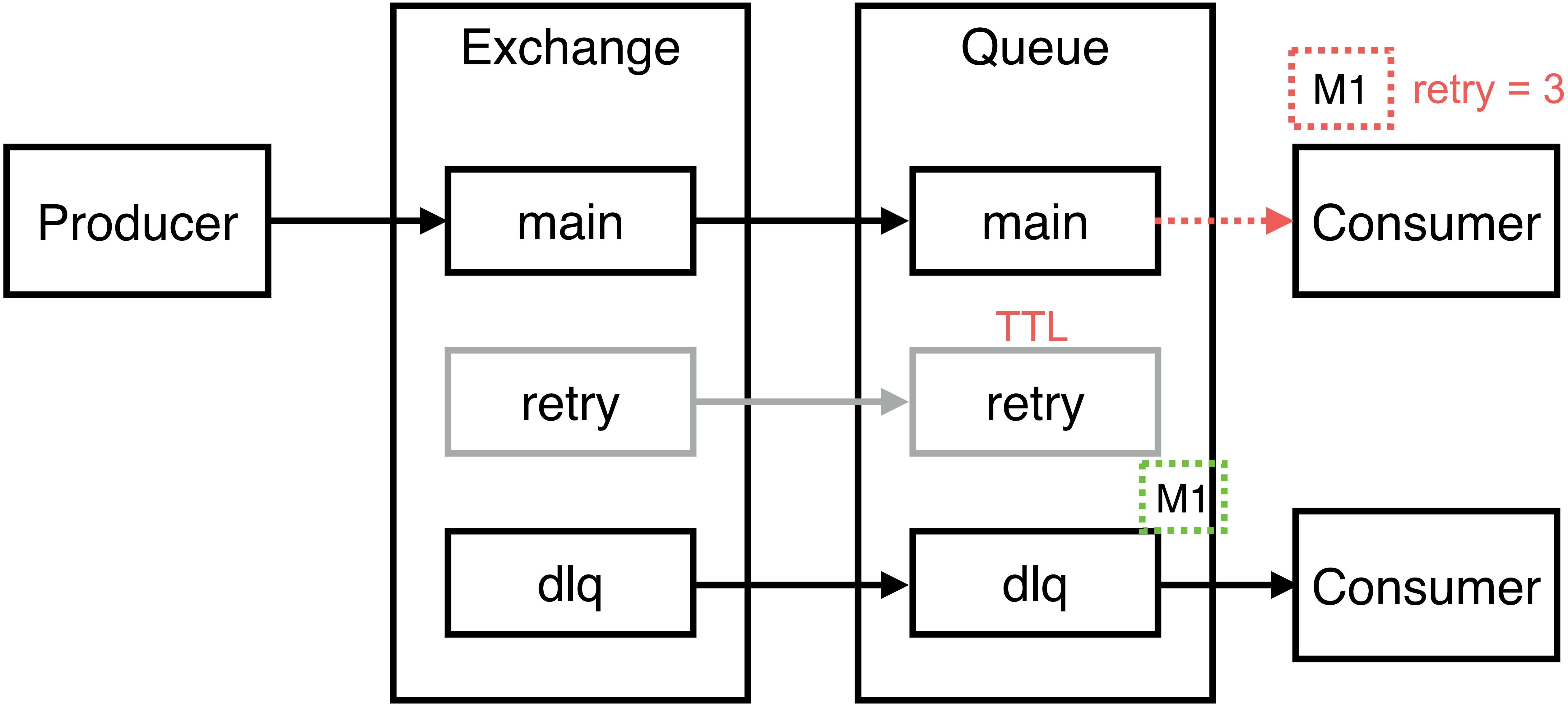
Retry and Dead Letter Queue

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



Retry and Dead Letter Queue

When retry > 3 then move message to **dlq** exchange

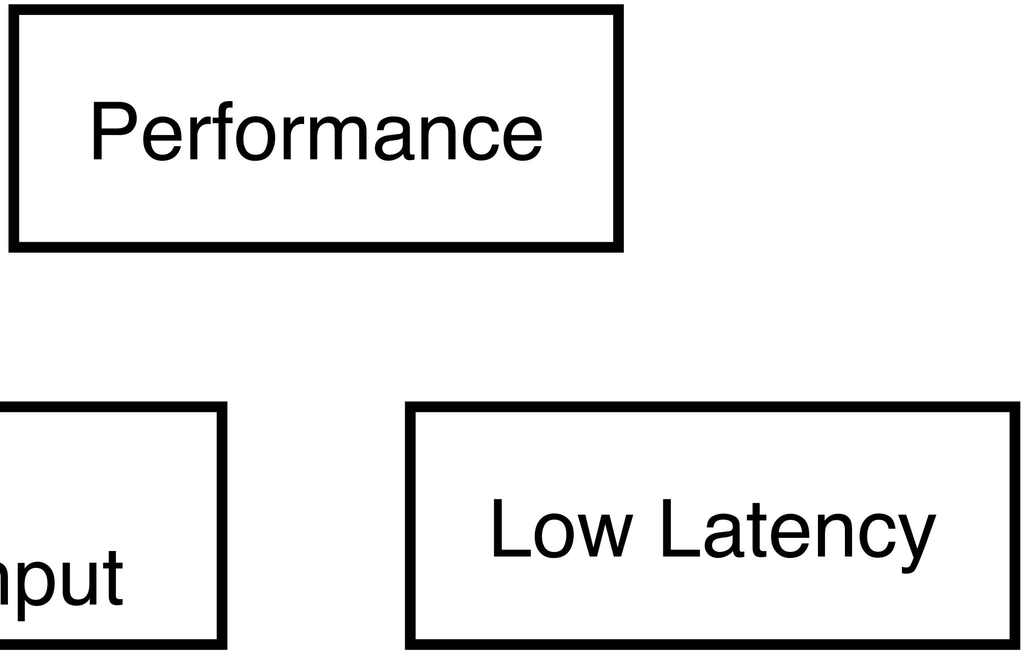


RabbitMQ Stream

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

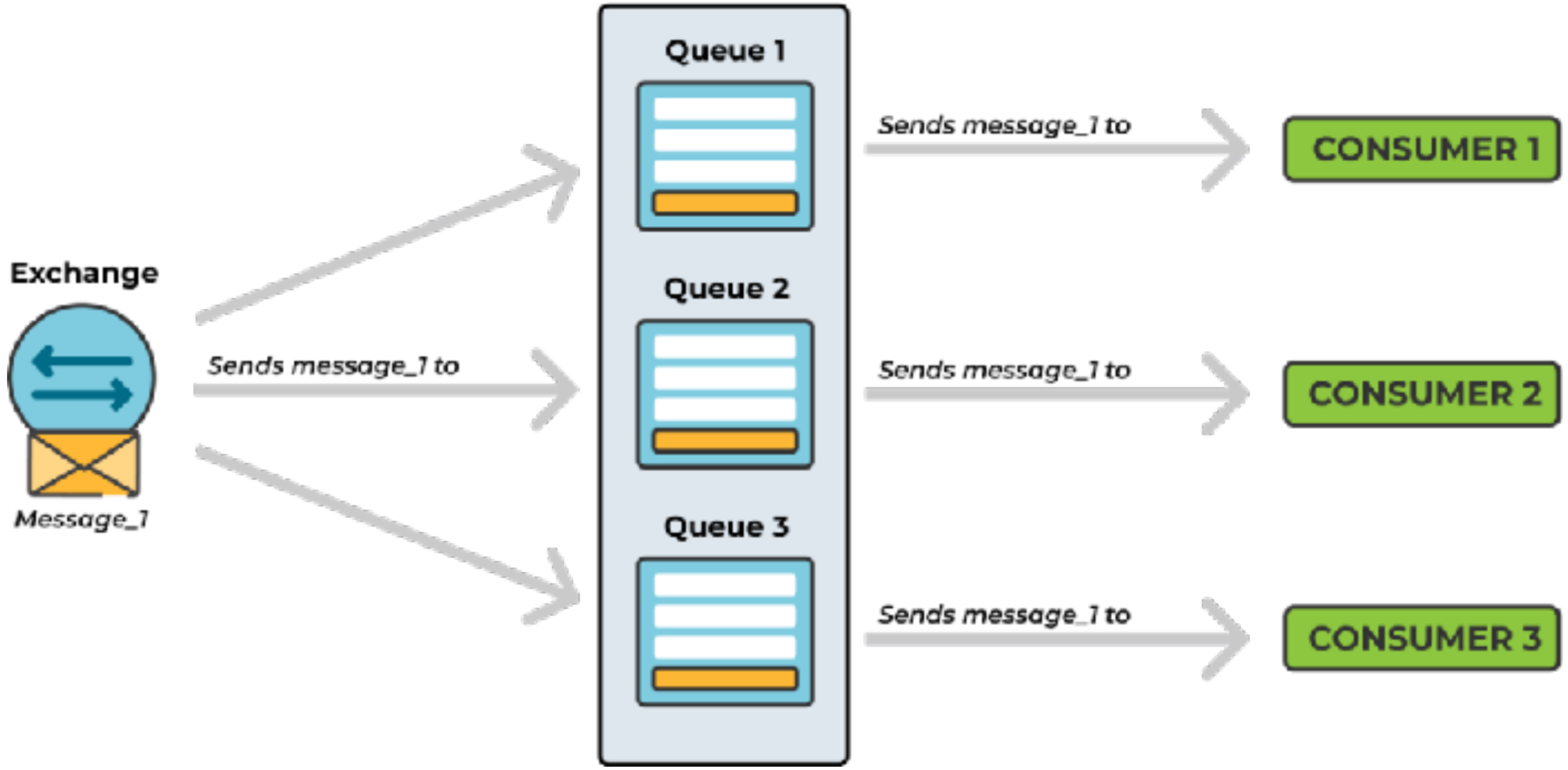
Queue vs Stream

Feature	Queue	Stream
Exclusivity	Yes	No
Non-durable queue	Yes	No
Message persistence	Per message	Always
TTL (Time To Live)	Yes	No (retention)
Dead letter queue	Yes	No
Queue length limit	Yes	No (retention)
Keep message in memory	Yes	Never
Message priority	Yes	No

https://www.rabbitmq.com/docs/streams#feature-matrix

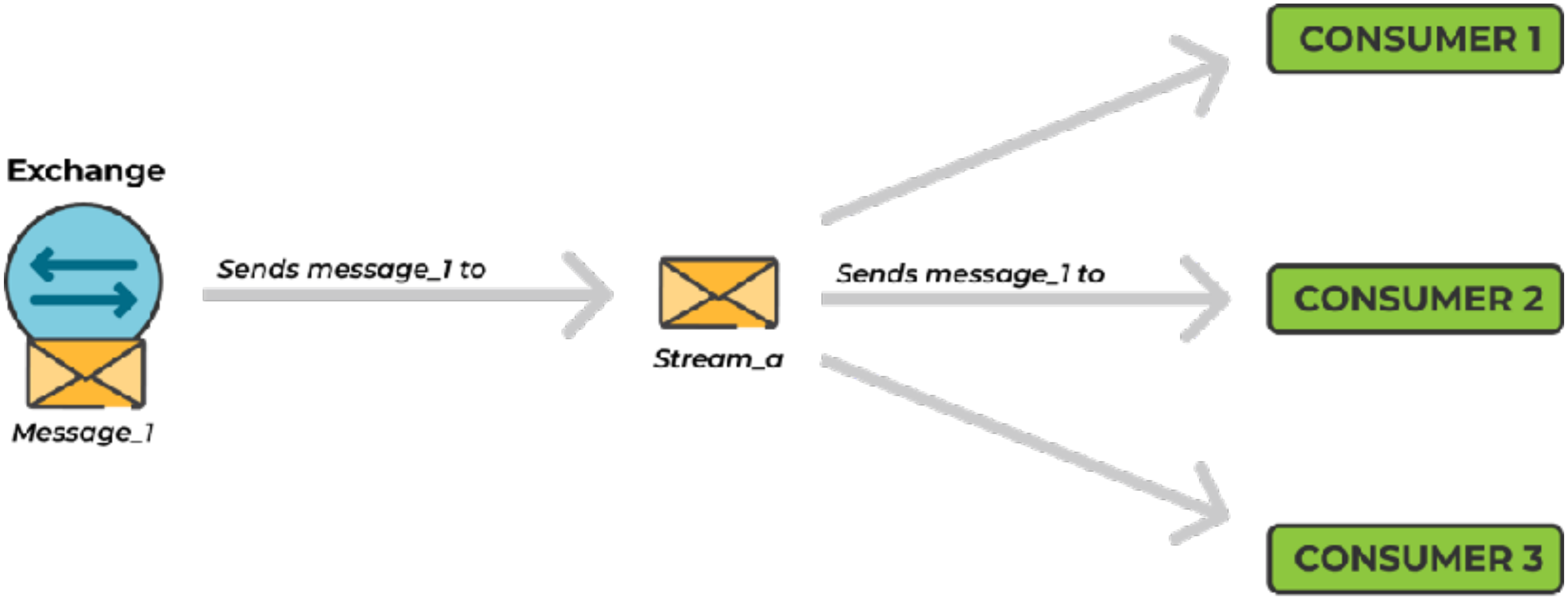
Fanout with Queue ?

Multiple consumers read the same message



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

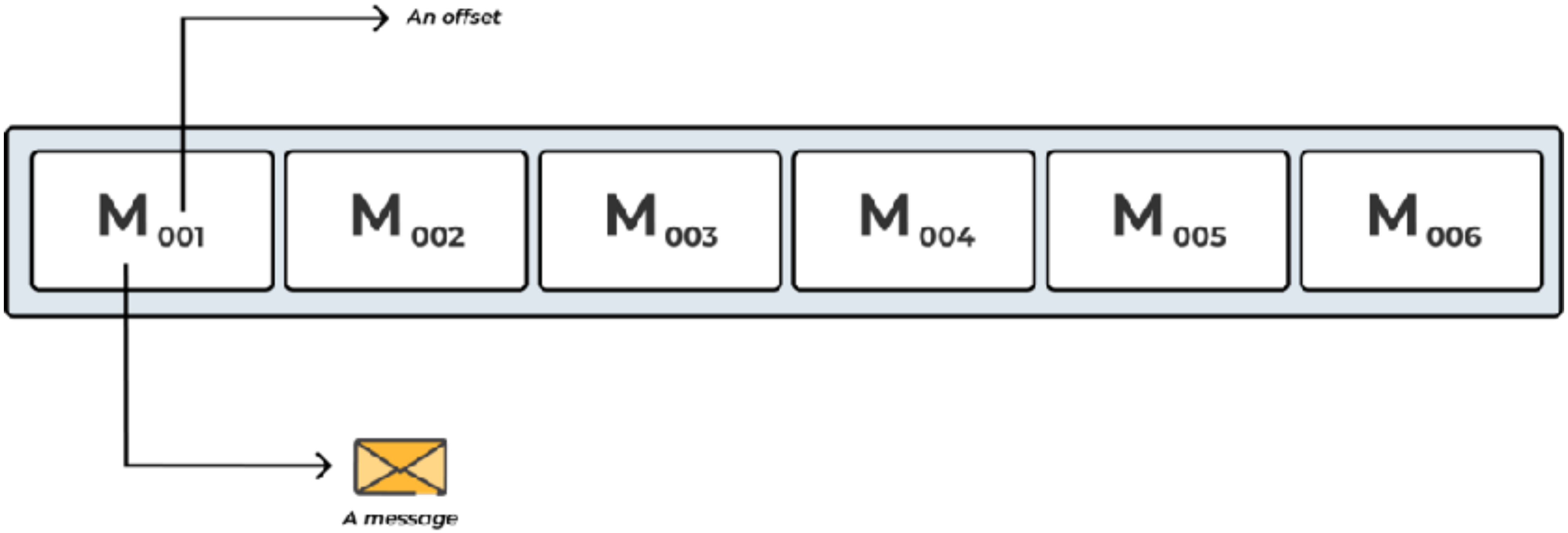
Fanout with Stream



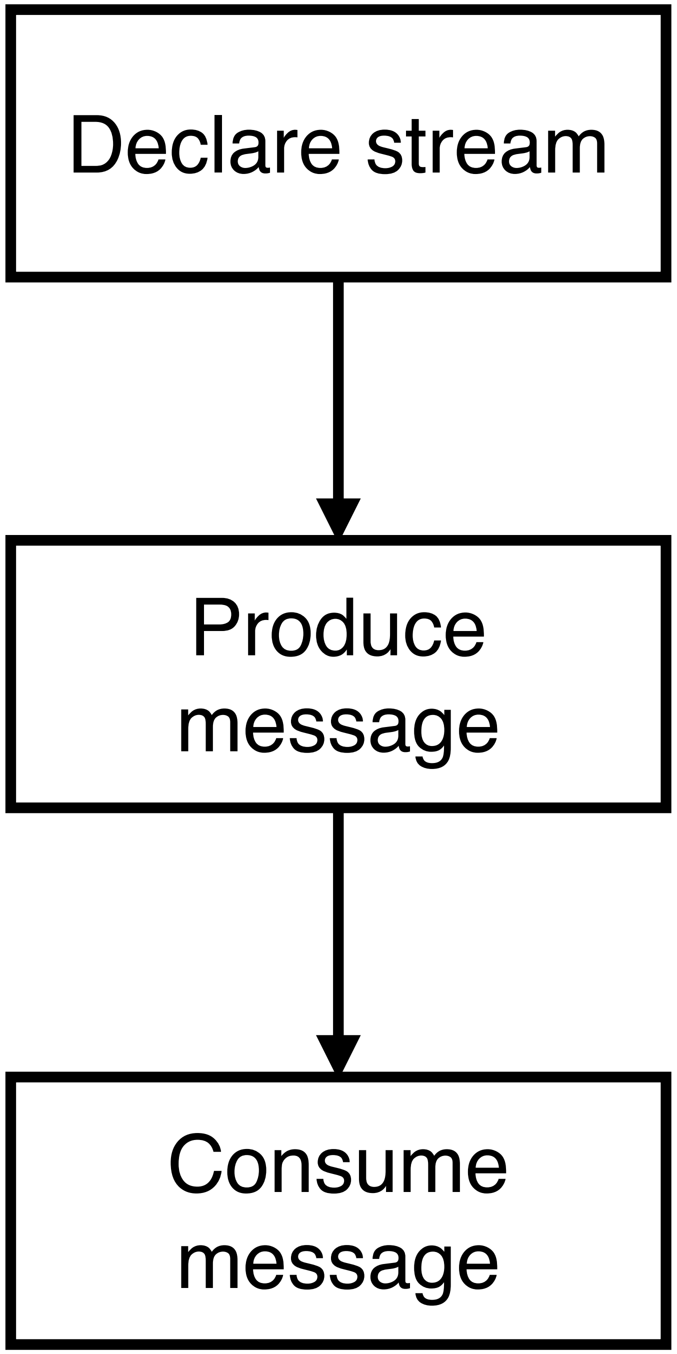
https://www.cloudamqp.com/blog/rabbitmq-streams-and-replay-features-part-1-when-to-use-rabbitmq-streams.html

Replay with offset

When consumer needs to re-read the same message



Steps to use Stream ?



Workshop
Fanout with Stream

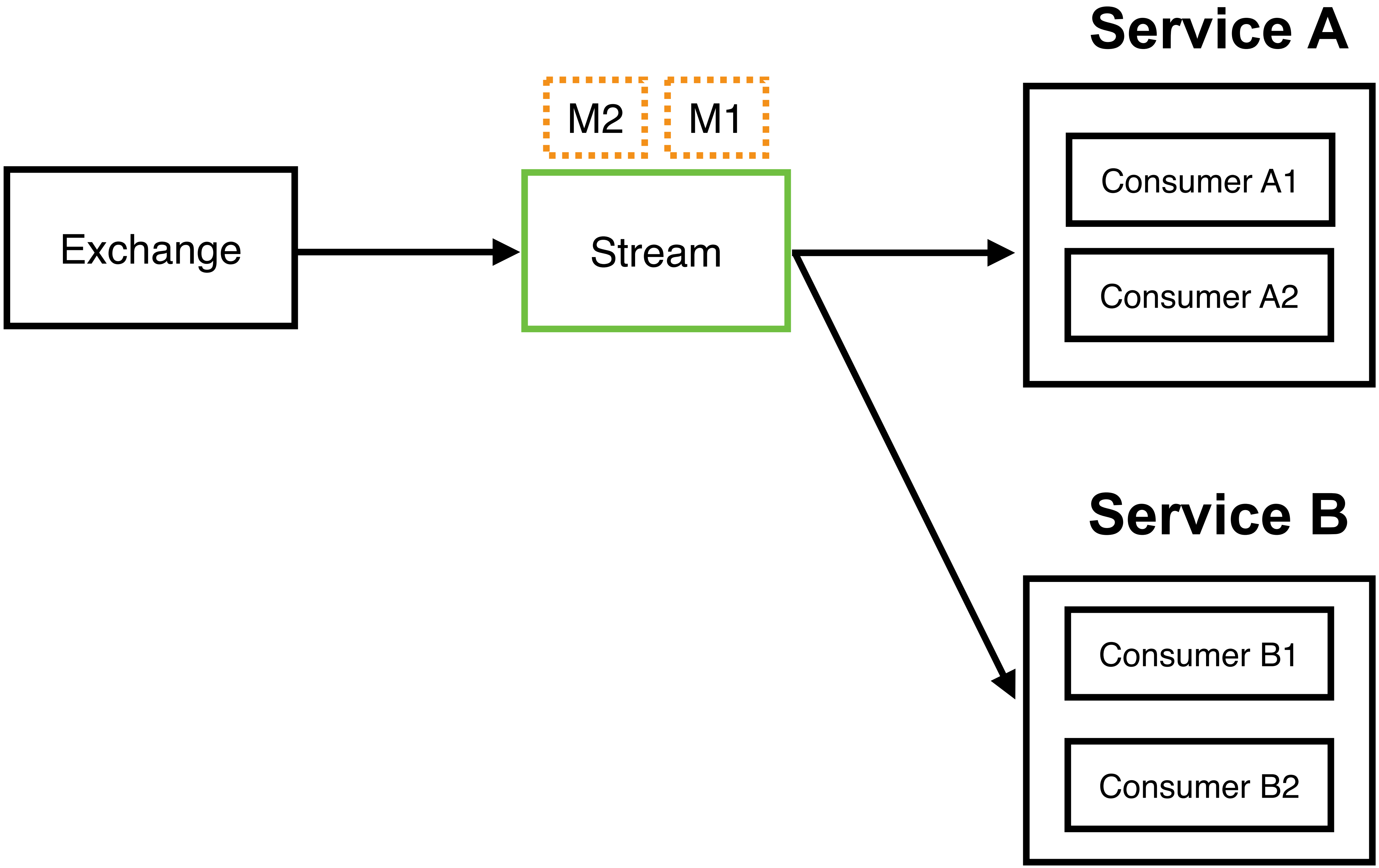
Fanout

Tracking offset

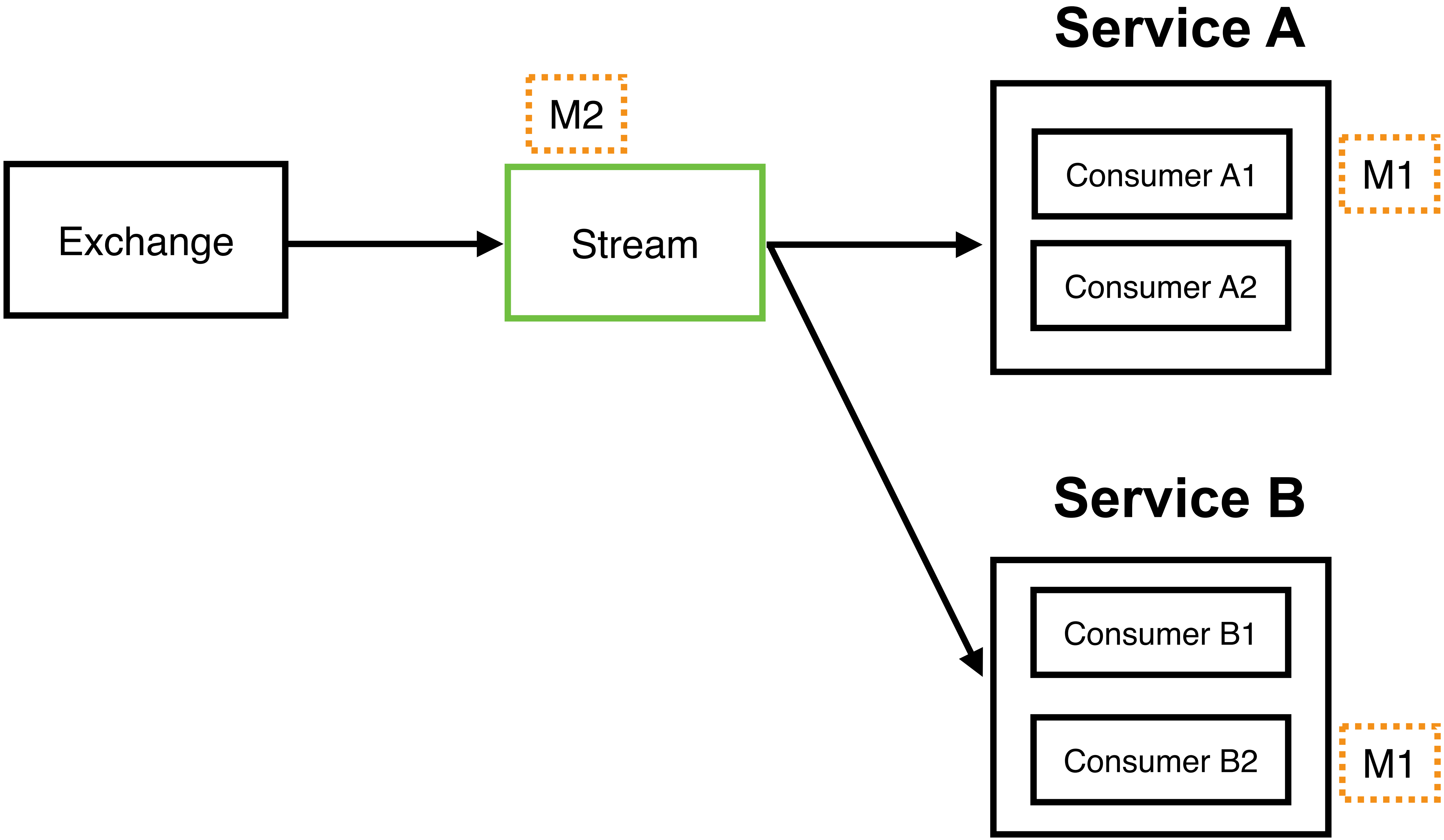
Filter message

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

Requirement (1)



Requirement (2)

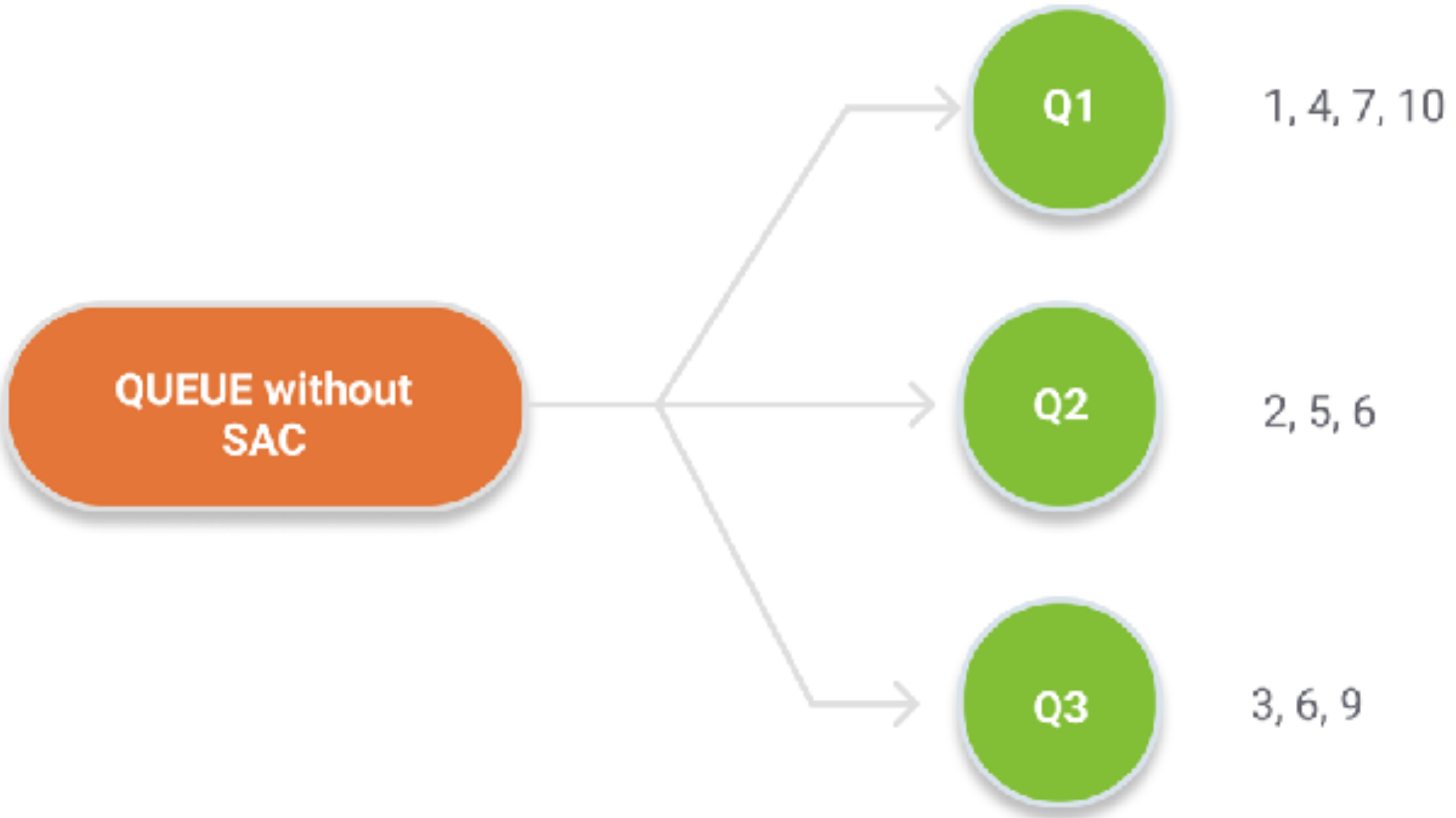


SAC
Single Active Consumer

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

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

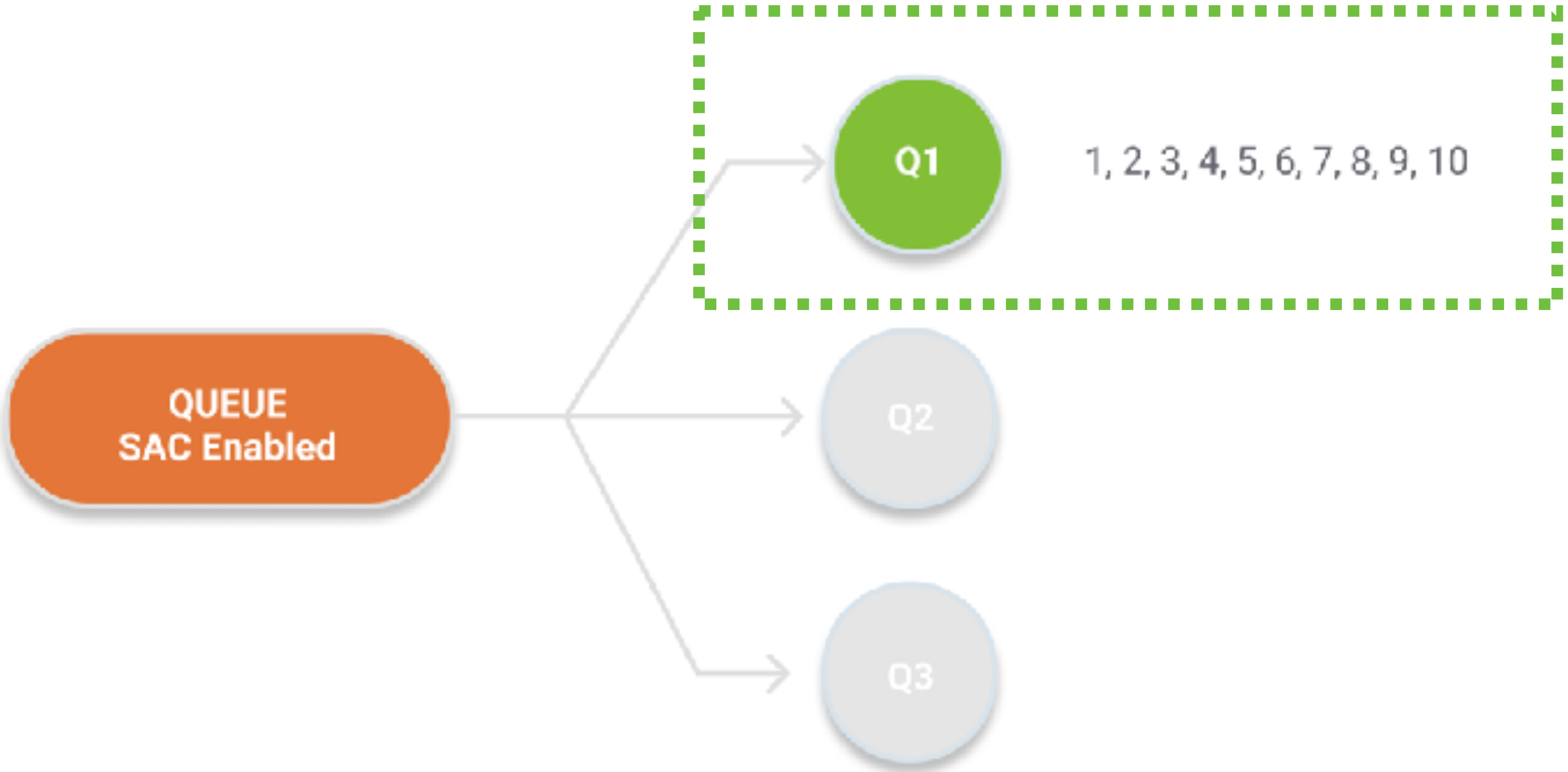
Queue without SAC



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



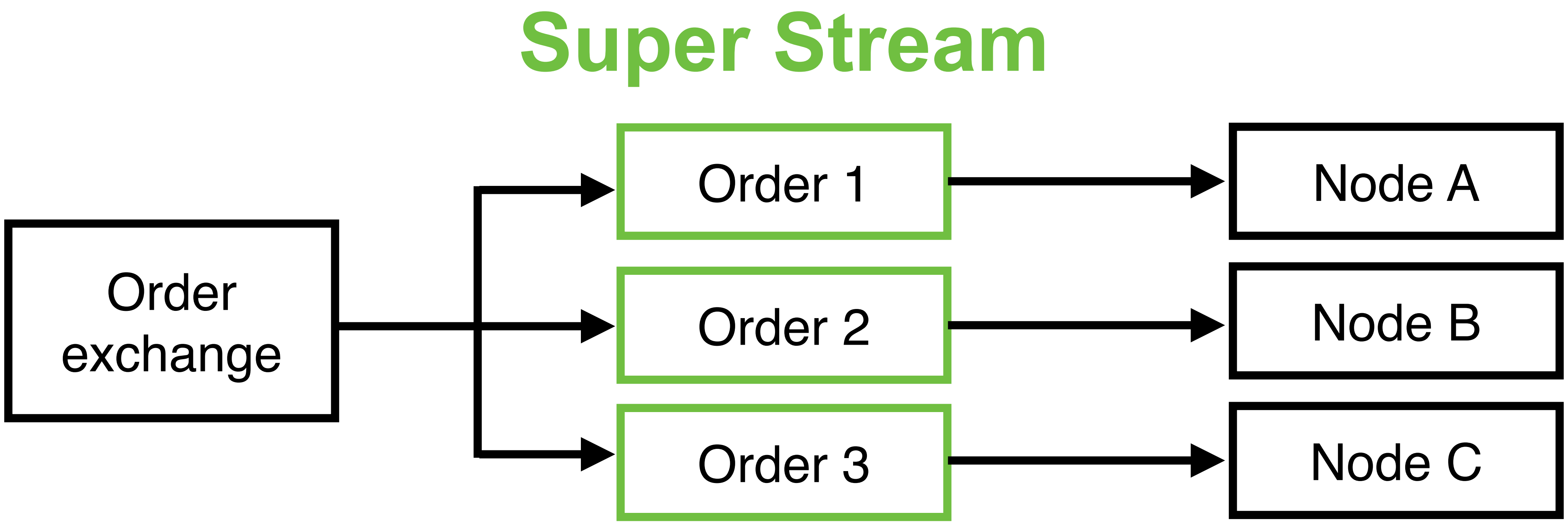
https://www.cloudamqp.com/blog/rabbitmq-3-8-feature-focus-single-active-consumer.html

Super Streams

(Partition the stream)

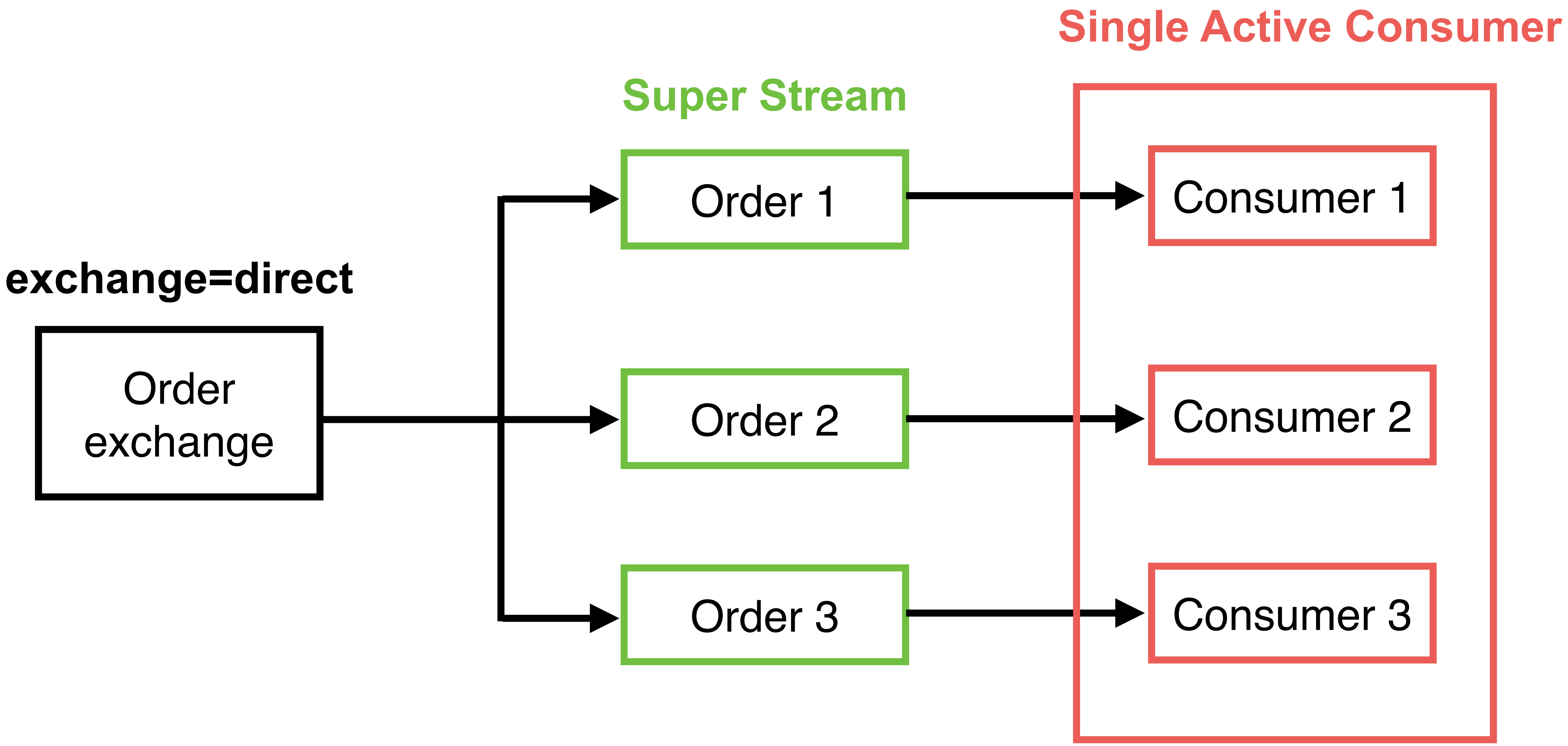
Super Streams ?

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



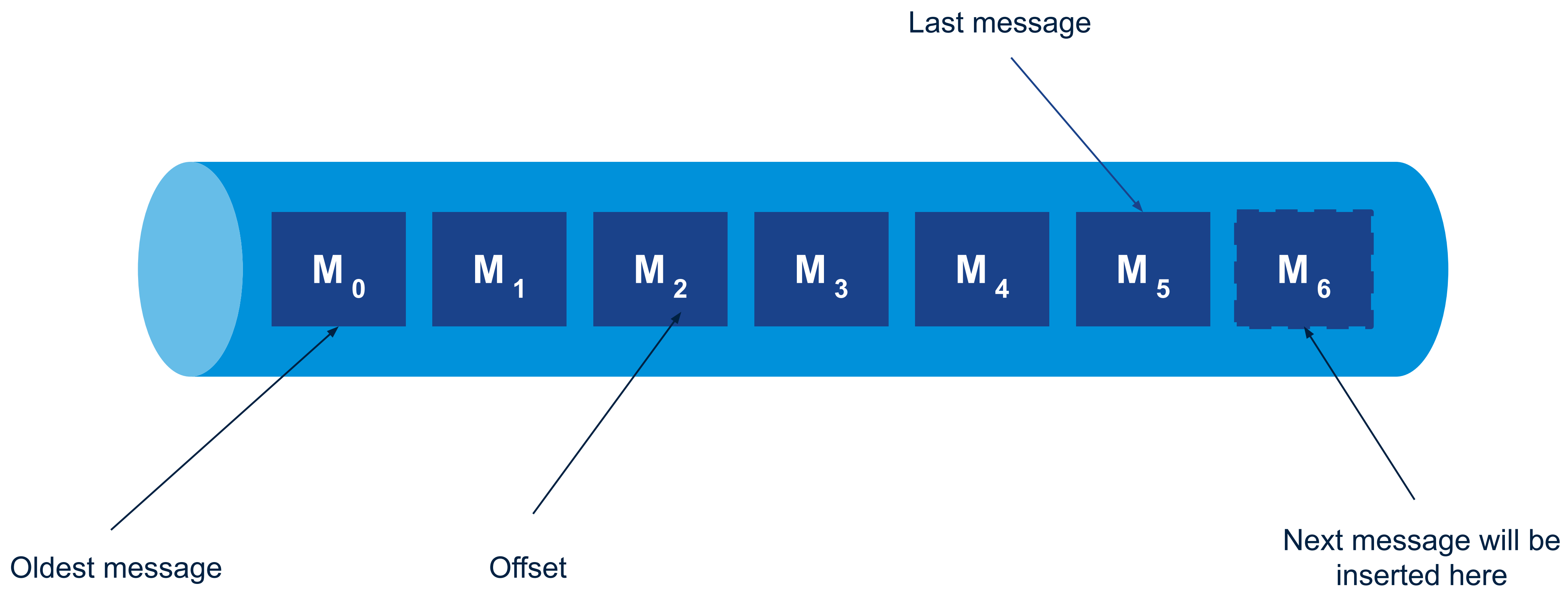
https://github.com/up1/course-rabbitmq-2025/tree/main/workshop/demo-go/demo-stream/basic

RabbitMQ Stream

Offset !!

https://www.rabbitmq.com/blog/2021/09/13/rabbitmq-streams-offset-tracking

Offset tracking



Last message

M_0 M_1 M_2 M_3 M_4 M_5 M_6

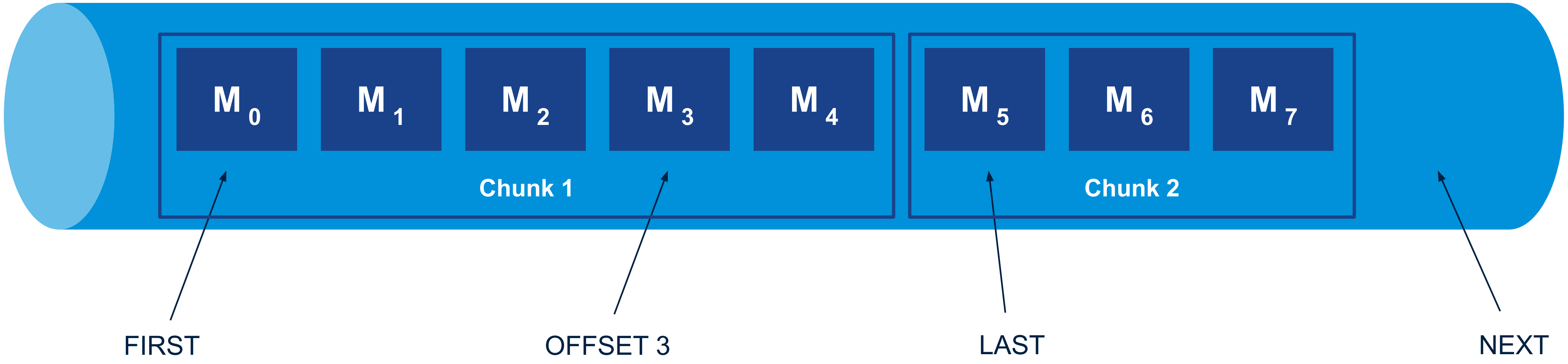
Oldest message

Offset

Next message will be inserted here

Offset tracking

Different offset in stream with 2 chunks



Q/A