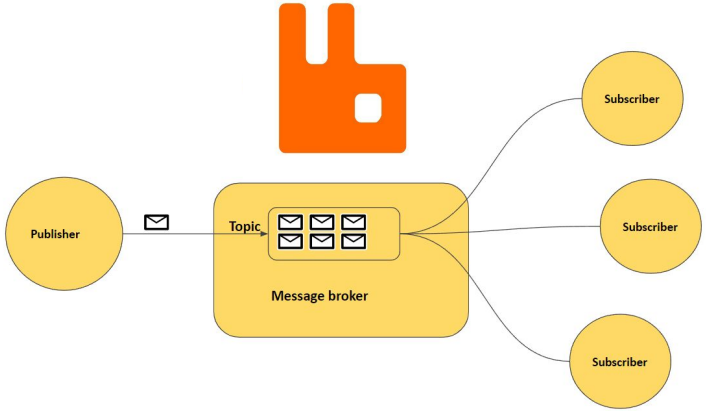
RabbitMQ for Newbies

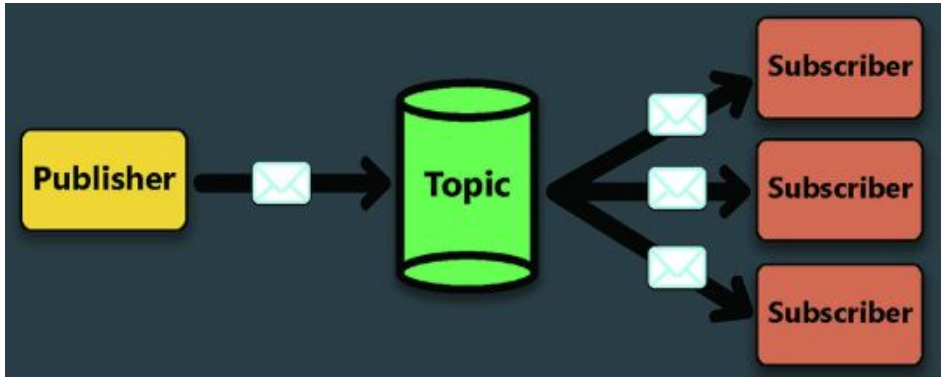
Playing with Pub/Sub Series

NetSIG Presentation



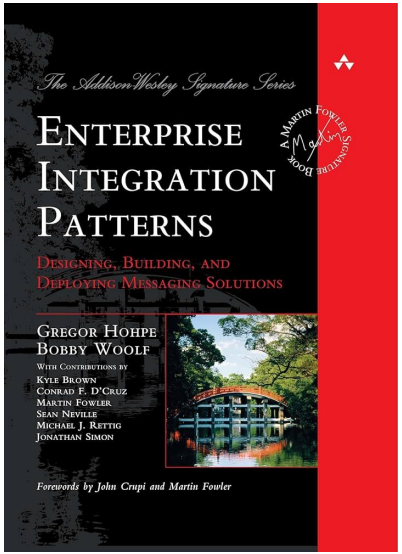
What is Pub/Sub?

- Pub/Sub = publish-subscribe.
- A messaging pattern for asynchronous communication.
- The pub/sub pattern decouples publishers from subscribers.
  - A pub/sub broker hosts a topic.
  - Subscribers register interest a topic.
  - Publishers produce messages on a topic.
  - Subscribers consume these messages.

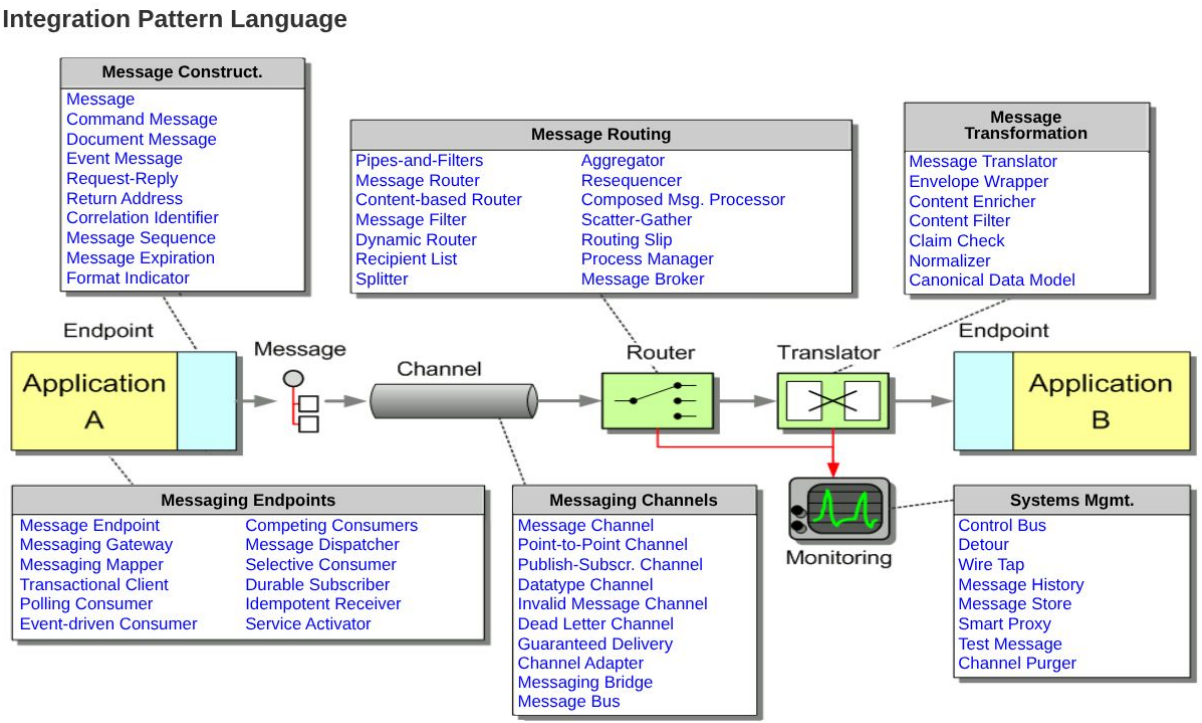


Messaging Architecture

- Enterprise Integration Patterns
  - Gregor Hohpe
  - Bobby Woolf
- The canon of messaging system design.
- Provides a common language.
- **<u>Core idea</u>**:
  - Use asynchronous messaging to decouple applications.

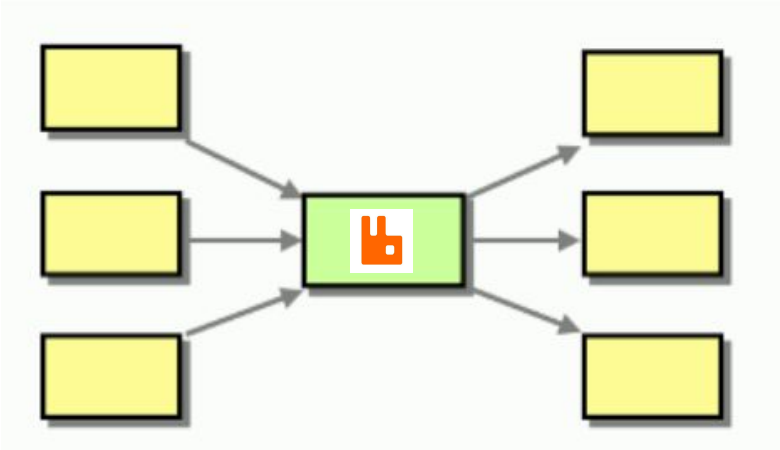


Architecture Components



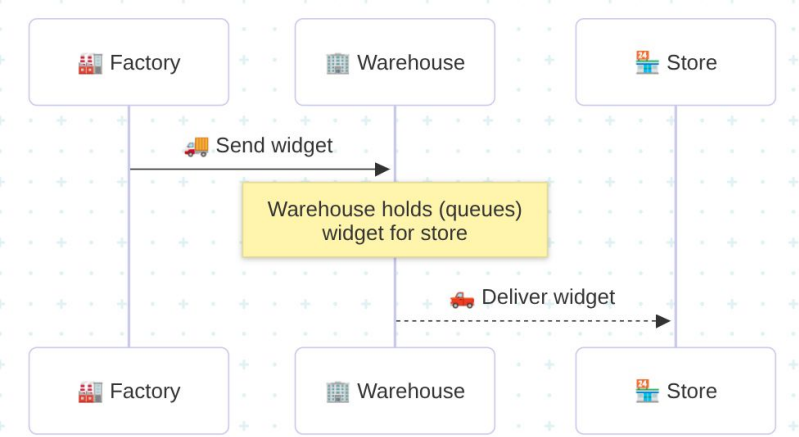
Message Broker

- RabbitMQ is our message broker.
- Central hub for messages routing.
- Follows a hub-and-spoke architecture.
- Enables centralized flow control.
- Broker decouples "publishers" from "subscribers".



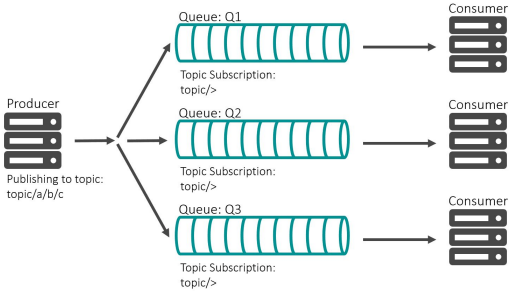
Asynchronous Communications

- Sender does not wait for receiver.
- Messages queued through the broker until processed.
- Enables loose coupling:
  - Improves reliability.
  - Improves scalability.
- Analogy : Warehouse shipping.



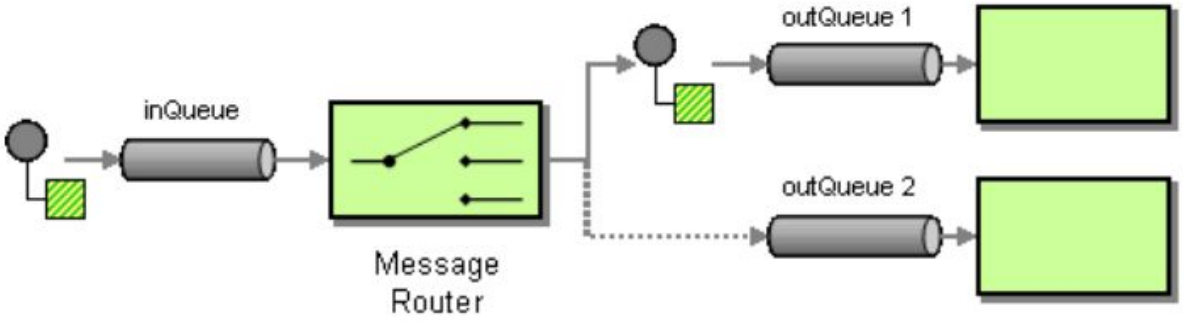
Topic

- Logical way for sorting messages.
- Publishers send message to a named topic.
- Subscribers get messages for that topic only.
- Enables one-to-many message delivery.
- Topic and queue are often used interchangeably.



Message Router (Exchange)

- Directs messages to correct destination.
- Uses rules, keys, or headers for routing.
- Decouples sender from destination logic.
- Supports point-to-point or pub-sub messaging.



RabbitMQ

RabbitMQ - Key Features

- Open-source message broker.
- Management UI + CLI for monitoring/control.
- Enables asynchronous communication between services.
- Plugins extend features - e.g. MQTT, Shovel, Prometheus, etc.
- Option to cluster for HA and scalability.
- Federation connects remote brokers over WAN.
- Lightweight & stable - Ideal solution for low-cost enterprise messaging.

RabbitMQ - UI

RabbitMQ™ RabbitMQ 4.1.4 Erlang 27.3.4.3

Refreshed 2025-10-13 15:22:31 Refresh every 5 seconds
Virtual host All
Cluster rabbit@fc1b9aabe61f
User admin Log out

Overview | Connections | Channels | Exchanges | Queues and Streams | Admin

Overview

Totals

Queued messages last minute ?

Ready	0
Unacked	0
Total	0

Message rates last minute ?

Publish	0.00/s	Deliver (auto ack)	0.00/s	Get (manual ack)	0.00/s	Unroutable (return)	0.00/s
Publisher confirm	0.00/s	Consumer ack	0.00/s	Get (auto ack)	0.00/s	Unroutable (drop)	0.00/s
Deliver (manual ack)	0.00/s	Redelivered	0.00/s	Get (empty)	0.00/s		

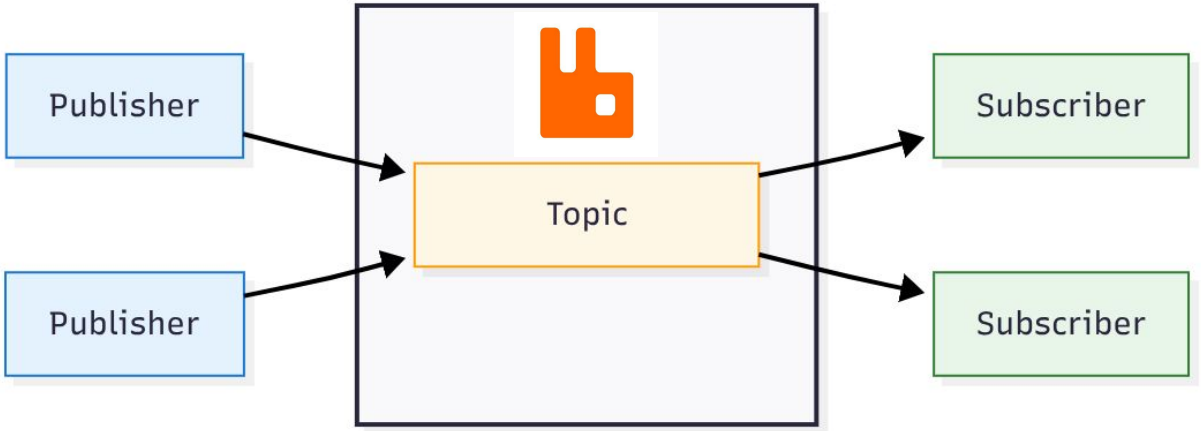
Global counts ?

Connections: 2 | Channels: 2 | Exchanges: 7 | Queues: 1 | Consumers: 1

Nodes

Name	File descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Cores	Info	Reset stats	+/-
rabbit@fc1b9aabe61f	40 1048576 available	450 1048576 available	123 MiB 9.2 GiB high watermark	24 GiB 48 MiB low watermark	1m 6s	8	basic 2 rss	This node All nodes	

MQTT



MQTT - Use Cases

- MQTT = Message Queuing Telemetry Transport
- Lightweight pub/sub messaging protocol. 📨
- Use Cases:
  - IoT sensor data ingestion. 🌡️
  - Mobile or edge device messaging. 📱
  - Telemetry streams. 📊

MQTT Demo - Local Docker Setup

- Note - Requires **`docker`** and **docker compose**

```
$ git clone \
https://github.com/netserf/netsig-presentation-rabbitmq-for-newbies.git
$ cd mqtt
$ make build
$ make up
$ make logs
```

- Browse to http://localhost:15672

  username: **`admin`**

  password: **`YodaSaysUseStrongPwd9!`**

MQTT Demo - CLI

Terminal config:

```
make exec-server
export MQTT_HOST=localhost
export MQTT_USER=admin
export MQTT_PASS=YodaSaysUseStrongPwd9!
alias msub='mosquitto_sub -h $MQTT_HOST -u $MQTT_USER -P $MQTT_PASS'
alias mpub='mosquitto_pub -h $MQTT_HOST -u $MQTT_USER -P $MQTT_PASS'
```

Terminal 1: Listen on dev/jokes topic

```
msub -t dev/jokes
```

Terminal 2: Publish to dev/jokes topic

```
mpub -t dev/jokes -m "It works on my machine!"
```

MQTT - Packet Capture

No.	Time	Source	Destination	Protocol	Length	Info
1	2025-10-16 12:24:35.055744	172.19.0.4	172.19.0.2	MQTT	176	Publish Message (id=6) [demo/topic]
3	2025-10-16 12:24:35.057003	172.19.0.2	172.19.0.4	MQTT	76	Publish Ack (id=6)
4	2025-10-16 12:24:35.057021	172.19.0.4	172.19.0.2	TCP	72	41157 → 1883 [ACK] Seq=105 Ack=5 Win=502

```
+ Frame 1: 176 bytes on wire (1408 bits), 176 bytes captured (1408 bits)
+ Linux cooked capture v2
+ Internet Protocol Version 4, Src: 172.19.0.4, Dst: 172.19.0.2
+ Transmission Control Protocol, Src Port: 41157, Dst Port: 1883, Seq: 1, Ack: 1, Len: 104
- MQ Telemetry Transport Protocol, Publish Message
  + [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet?)]
  + Header Flags: 0x32, Message Type: Publish Message, QoS Level: At least once delivery (Acknowledged deliver)
    Msg Len: 102
    Topic Length: 10
    Topic: demo/topic
    Message Identifier: 6
    Message: 4d65737361676520233620667 26f6d2070726f6475636572206174203230323 52d31302d…
```

```
0000  08 00 00 00 00 00 00 18  00 01 03 06 c2 21 3e fd   ···· ···· ·····!>·
0010  84 d6 00 00 45 00 00 9c  c0 c4 40 00 40 06 21 6b   ····E··· ··@·@·!k
0020  ac 13 00 04 ac 13 00 02  a0 c5 07 5b d7 b0 c0 7a   ····· ···[···z
0030  b7 8f f0 e9 80 18 01 f6  58 bb 00 00 01 01 08 0a   ····· X·····
0040  75 d9 86 2c 39 b9 a8 8f  32 66 00 0a 64 65 6d 6f   u··,9··· 2f··demo
0050  2f 74 6f 70 69 63 00 06  4d 65 73 73 61 67 65 20   /topic·· Message
0060  23 36 20 66 72 6f 6d 20  70 72 6f 64 75 63 65 72   #6 from  producer
0070  20 61 74 20 32 30 32 35  2d 31 30 2d 31 36 54 31    at 2025 -10-16T1
0080  39 3a 32 34 3a 33 35 2e  30 35 35 32 39 31 20 3a   9:24:35. 055291 :
0090  20 5b 27 68 69 70 27 2c  20 27 68 69 70 27 5d 20    ['hip',  'hip'] 
00a0  28 68 69 70 20 68 69 70  20 61 72 72 61 79 21 29   (hip hip  array!)
```

MQTT - Pros & Cons

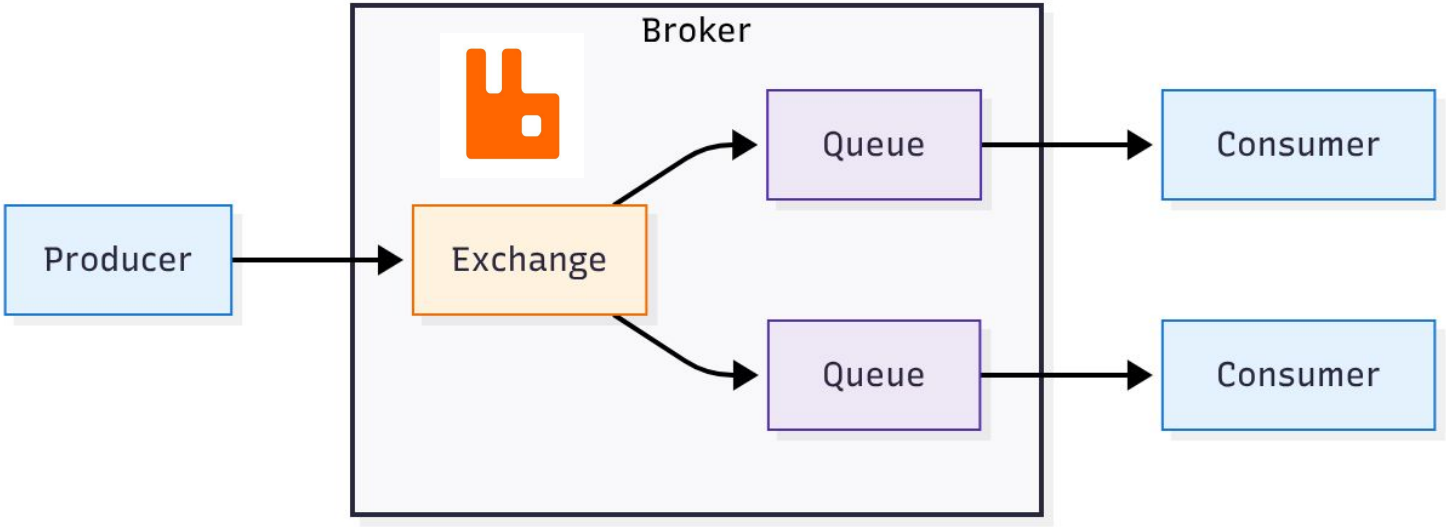
Pros

- ✔ Lightweight messaging protocol 🌐
- ✔ Good for low bandwidth networks 📶
- ✔ Supports many-to-many pub/sub 🔁

Cons

- ✖ Weak delivery guarantees vs AMQP 🚫
- ✖ Minimal built-in security 🔒

AMQP



AMQP Demo - Local Docker Setup

- Note - Requires **`docker`** and **docker compose**

```
$ git clone \
https://github.com/netserf/netsig-presentation-rabbitmq-for-newbies.git
$ cd amqp
$ make build
$ make up
$ make logs
```

- Browse to Browse to http://localhost:15672

  username: **`admin`**

  password: **`YodaSaysUseStrongPwd9!`**

AMQP Demo - CLI

Terminal config:

```
make exec-server
export RABBITMQ_USER=admin
export RABBITMQ_PASS=YodaSaysUseStrongPwd9!
alias rma='rabbitmqadmin -u $RABBITMQ_USER -p $RABBITMQ_PASS'
```

Create exchange:

```
rma declare exchange name=fruit type=direct
```

Create queues:

```
rma declare queue name=banana
rma declare queue name=apple
```

Create bindings:

```
rma declare binding source=fruit destination=banana routing_key=yellow
rma declare binding source=fruit destination=apple routing_key=red
```

AMQP Demo - CLI

```
make exec-server
export RABBITMQ_USER=admin
export RABBITMQ_PASS=YodaSaysUseStrongPwd9!
alias rma='rabbitmqadmin -u $RABBITMQ_USER -p $RABBITMQ_PASS'
```

Send messages with routing keys:

```
rma publish exchange=fruit routing_key=yellow payload="Mañana banana!"
rma publish exchange=fruit routing_key=red payload="Apple-y ever after!"
```

Get banana messages:

```
rma get queue=banana ackmode=ack_requeue_false
```

Get apple messages:

```
rma get queue=apple ackmode=ack_requeue_false
```

AMQP - Use Cases

- AMQP = Advanced Message Queuing Protocol.
- Default messaging protocol for RabbitMQ. 📨
- Use Cases:
  - Enterprise app integration. 🏢
  - Microservice communication. 🔄
  - Task & job queuing. ⚙️
  - Reliable event routing and workflows. 🧭

AMQP - Pros & Cons

Pros

- ✔ Supports both point-to-point and publish-subscribe messages. 📨
- ✔ Flexible routing options to direct messages. 🔄
- ✔ Security - Authentication / authorization.
- ✔ Rich Features - Built-in dead-lettering, TTL, and clustering. 🛠️

Cons

- ✖ Steeper learning curve than MQTT. 🤔
- ✖ Higher network overhead per message. 📜

AMQP - Packet Capture

No.	Time	Source	Destination	Protocol	Length	Info
1	2025-10-16 12:41:29.053315	172.19.0.3	172.19.0.2	AMQP	100	Basic.Publish x= rk=jokes_queue
2	2025-10-16 12:41:29.053356	172.19.0.2	172.19.0.3	TCP	72	5672 → 47884 [ACK] Seq=1 Ack=29 Win=50
3	2025-10-16 12:41:29.053397	172.19.0.3	172.19.0.2	AMQP	95	Content-Header
4	2025-10-16 12:41:29.053408	172.19.0.2	172.19.0.3	TCP	72	5672 → 47884 [ACK] Seq=1 Ack=52 Win=50
5	2025-10-16 12:41:29.053428	172.19.0.3	172.19.0.2	AMQP	203	Content-Body
6	2025-10-16 12:41:29.053438	172.19.0.2	172.19.0.3	TCP	72	5672 → 47884 [ACK] Seq=1 Ack=183 Win=5

```
+ Frame 5: 203 bytes on wire (1624 bits), 203 bytes captured (1624 bits)
+ Linux cooked capture v2
+ Internet Protocol Version 4, Src: 172.19.0.3, Dst: 172.19.0.2
+ Transmission Control Protocol, Src Port: 47884, Dst Port: 5672, Seq: 52, Ack: 1, Len: 131
- Advanced Message Queueing Protocol
    Type: Content body (3)
    Channel: 1
    Length: 123
    Payload: 4a6f6b65202331322061742032303235 2d31302d31365431393a34313a32392e30353239…
```

```
0000  08 00 00 00 00 00 00 1e  00 01 03 06 8a f0 d1 c2   ........ ........
0010  fb 8e 00 00 45 00 00 b7  69 73 40 00 40 06 78 a2   ····E··· is@·@·x·
0020  ac 13 00 03 ac 13 00 02  bb 0c 16 28 f9 44 2b 17   ........ ···(·D+·
0030  3d e8 2b c6 80 18 01 f5  58 d5 00 00 01 01 08 0a   =·+····· X·......
0040  24 a4 a0 1a d7 1b 57 88  03 00 01 00 00 00 7b 4a   $·······W· ......{J
0050  6f 6b 65 20 23 31 32 20  61 74 20 32 30 32 35 2d   oke #12  at 2025-
0060  31 30 2d 31 36 54 31 39  3a 34 31 3a 32 39 2e 30   10-16T19 :41:29.0
0070  35 32 39 36 32 3a 20 57  68 61 74 20 64 6f 65 73   52962: W hat does
0080  20 27 45 6d 61 63 73 27  20 73 74 61 6e 64 20 66    'Emacs'  stand f
0090  6f 72 3f 20 27 45 78 63  6c 75 73 69 76 65 6c 79   or? 'Exc lusively
00a0  20 75 73 65 64 20 62 79  20 6d 69 64 64 6c 65 20    used by  middle
00b0  61 67 65 64 20 63 6f 6d  70 75 74 65 72 20 73 63   aged com puter sc
00c0  69 65 6e 74 69 73 74 73  2e 27 ce                  ientists .'·
```

RabbitMQ Summary

- Bridges backend services (AMQP) & IoT devices (MQTT). 🌐
- A mature, battle-tested platform with a large plugin ecosystem. 🛠️
- Decouples applications, enabling reliable message delivery at scale. 🔄

Resources

- **Main RabbitMQ Site** - https://www.rabbitmq.com/
- **RabbitMQ on GitHub** - https://github.com/rabbitmq
- **RabbitMQ Docs** - https://www.rabbitmq.com/docs
- **Enterprise Integration Patterns Site -** https://www.enterpriseintegrationpatterns.com/
- **Presentation Slides & Demo** - https://github.com/netserf/netsig-presentation-rabbitmq-for-newbies

Questions

Possible Future Discussions

- Orchestration
  - Nomad
- Workflow Automation
  - n8n
- Pub/Sub Series
  - Kafka
  - GCP Pub/Sub
- Monitoring
  - Prometheus / Grafana
  - Consul
  - Loki
  - osquery
- CI / CD
  - GitHub Actions
  - Woodpecker CI