101 ways to break your RabbitMQ

~$ whoami

Denis GERMAIN

Senior SRE at deezer

Tech blogger blog.zwindler.fr*

@zwindler / @zwindler_rflx

#geek #SF #running

** Slides are available on the blog*

101 ways to break your RabbitMQ

A little bit of context

A little bit of context (2)

- Historically on-prem monoliths
- Start again, from scratch
  - Migrate to cloud SaaS
  - Switch to microservices
- How can we secure the messages these µ-services (and on-prem equipments) exchange?
  - RabbitMQ

Why RabbitMQ?

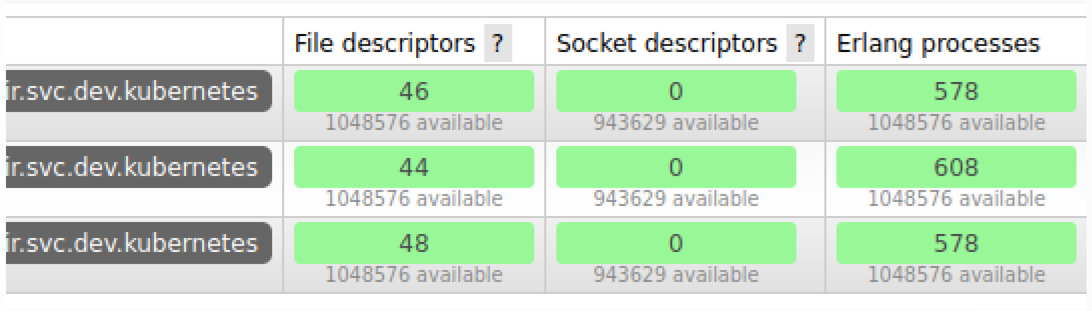
- Broad range of supported protocols
- Extra layer of abstraction with « Exchanges »
- Clustering, high availability and replication features
- Management UI and REST API
- On-the-fly configuration

What could possibly go wrong ?

Let's break RabbitMQ!

Installation & design

OS parameters

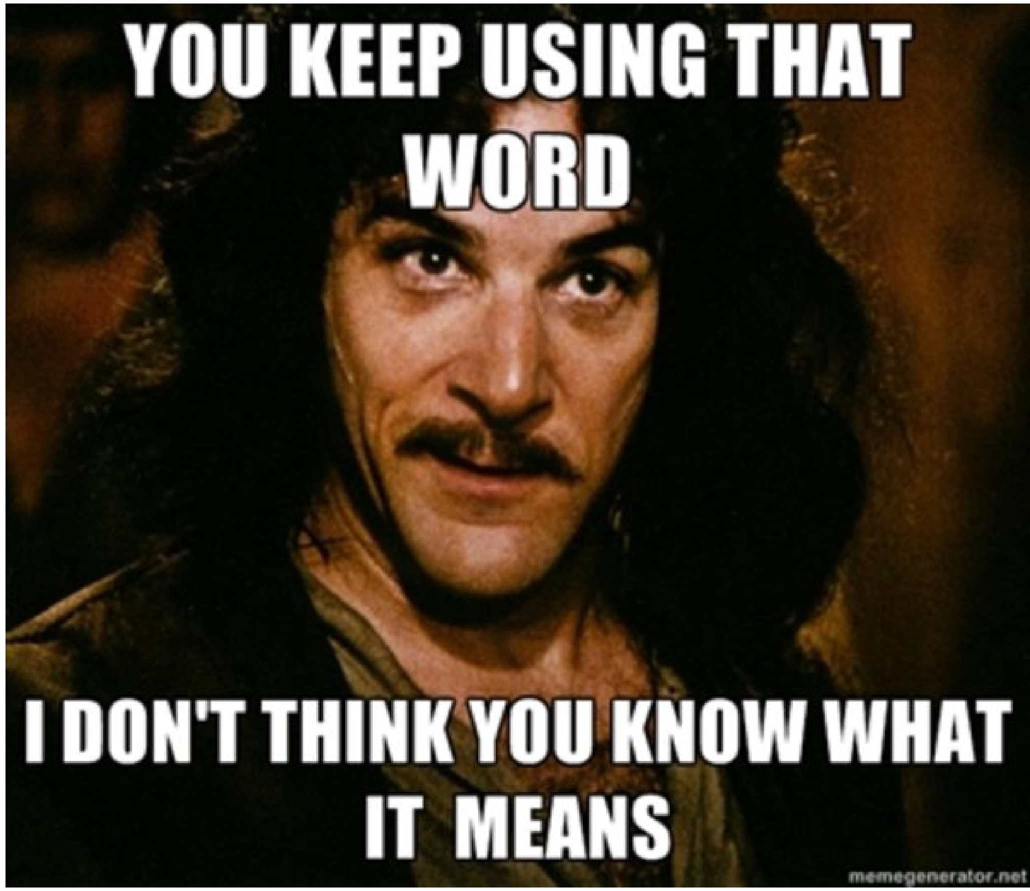


	File descriptors ?	Socket descriptors ?	Erlang processes
ir.svc.dev.kubernetes	46 1048576 available	0 943629 available	578 1048576 available
ir.svc.dev.kubernetes	44 1048576 available	0 943629 available	608 1048576 available
ir.svc.dev.kubernetes	48 1048576 available	0 943629 available	578 1048576 available

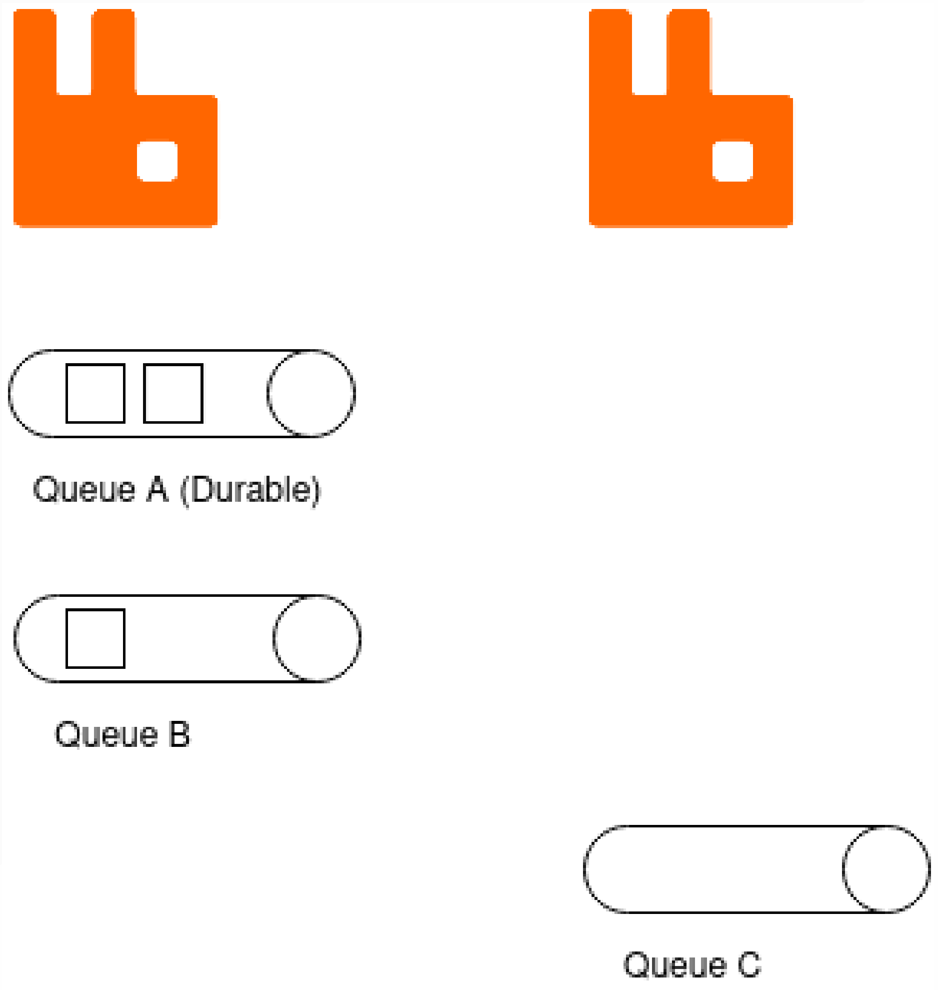
- N°1 advice from Michael Klishin's "RabbitMQ Operations" from 2015 (and still true)
- Bump your file descriptors! (really)
- Other fine tuning network parameters might be useful

I want 0 downtime, let's put RabbitMQ in cluster!

- Nodes share all logical objects *except queues*
- **Only one node** owns a queue
  - Nodes know which queue is owned by which node
  - Messages are transparently redirected to the right node



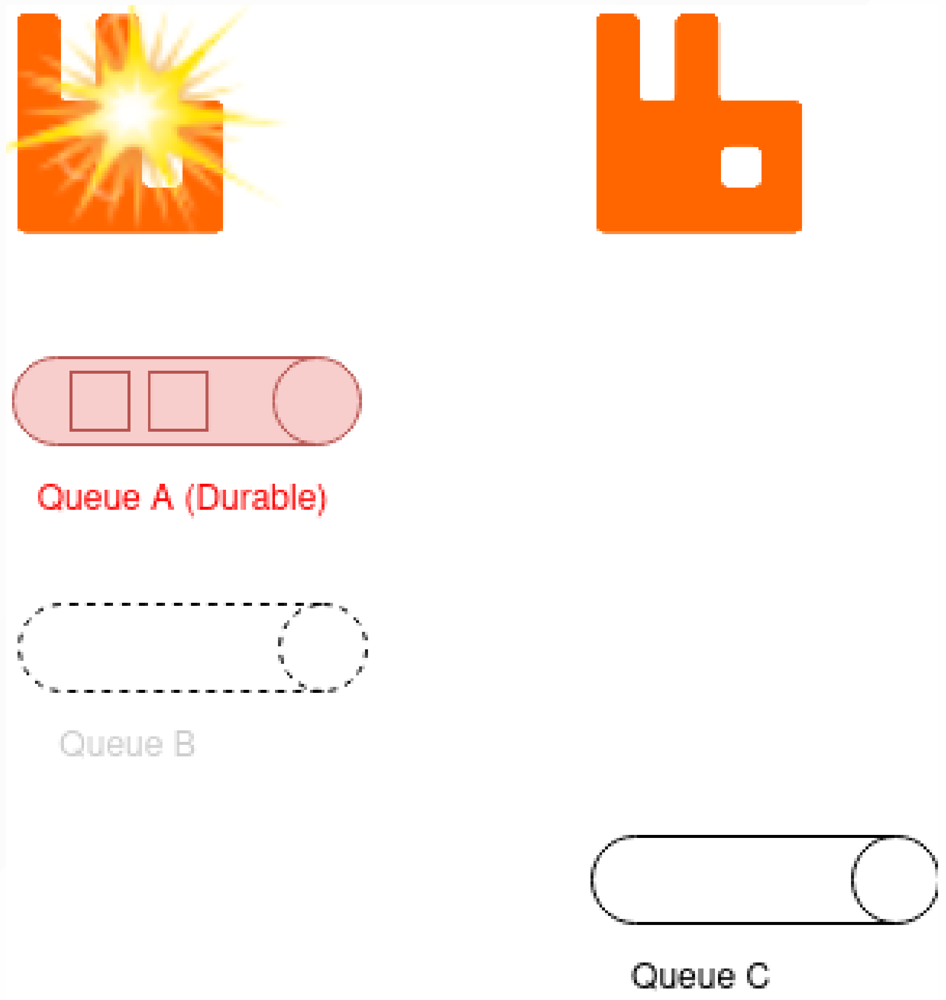
RabbitMQ cluster



Queues are not magically replicated

- Nodes are still SPOF 😱
- Queues owned by a downed node become unavailable
  - Non-durable queues are destroyed
  - **Durable** queues are blocked

RabbitMQ cluster



What you *really* want when thinking clusters

2 strategies:

- Don't configure queues as **durable**, retry and redeclare
- Flag some queues as « highly available »
  - "classic" HA queues (1 leader + promotable mirrors)
  - Quorum queues (3.8+, raft consensus)

Beware of the HA queues and parameters

HA queues :

- need more RAM
- latency **++** & throughput **--**
- multiple modes (with implication)
  - ha-mode (how many mirrors)
  - ha-sync-mode (may block queues)

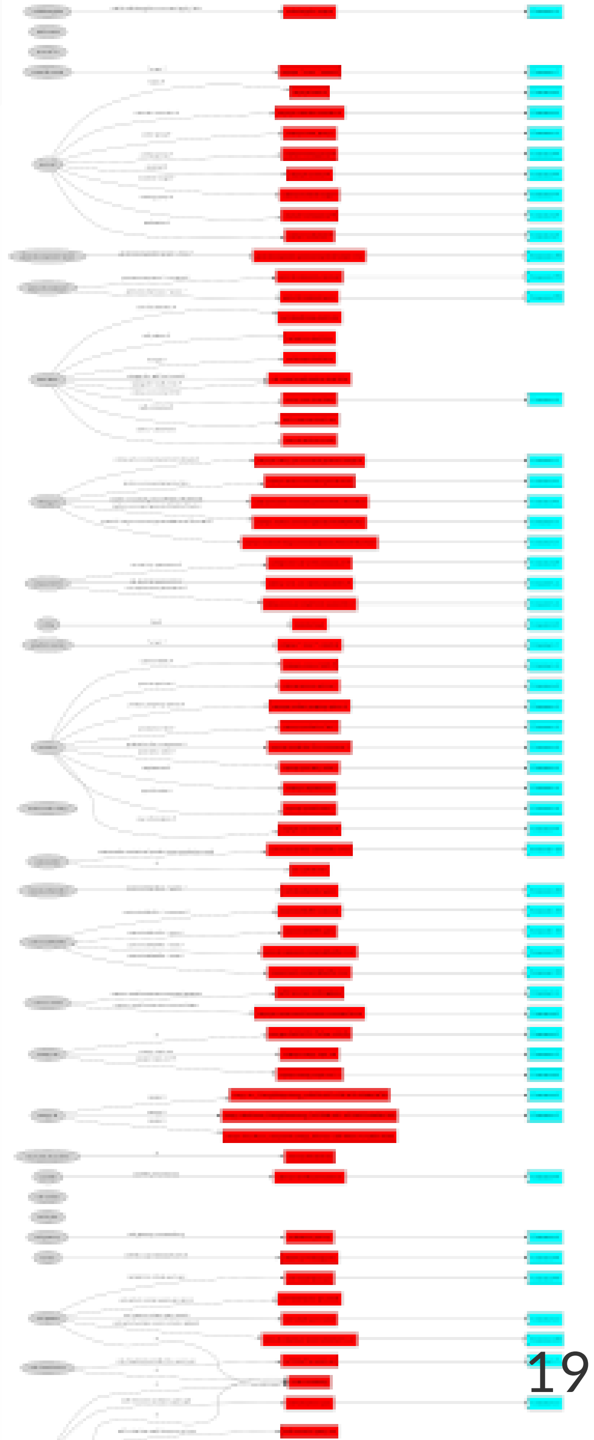
Use odd number of nodes

- Network partitions happen
- Clusters with even number of nodes theoretically possible
- Save yourself a world of pain
  - use only odd number of nodes

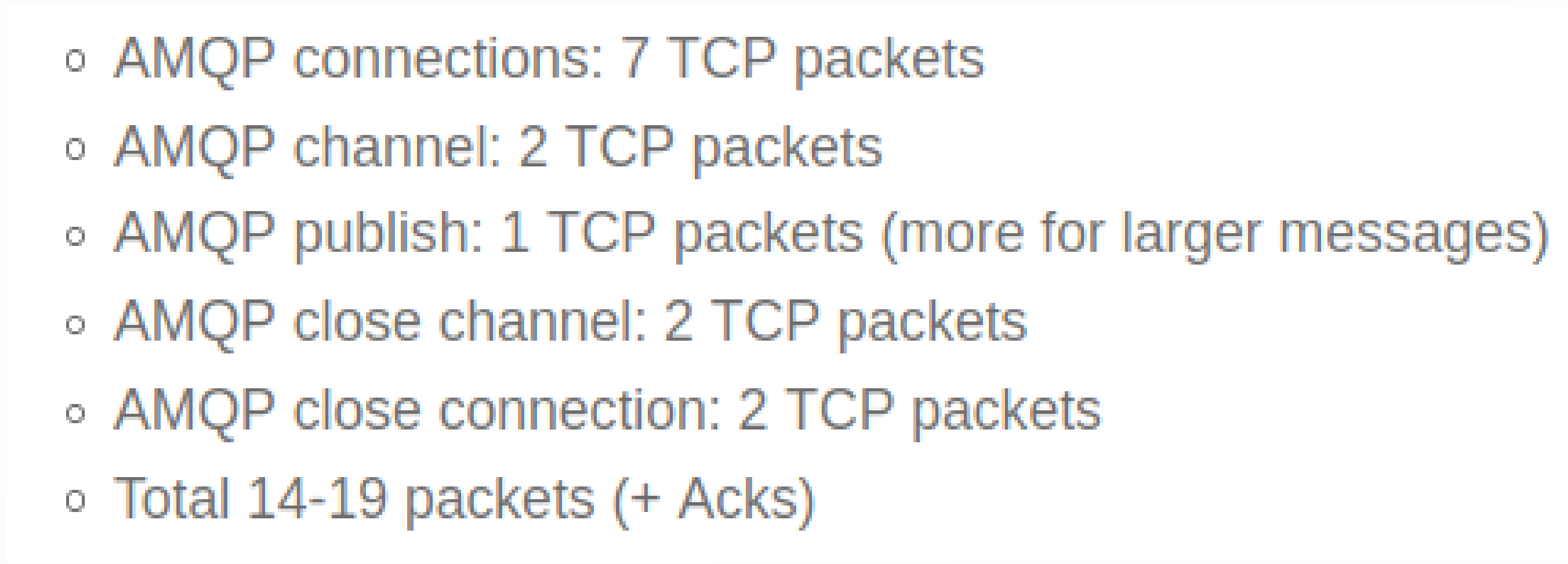
RabbitMQ usage

Security

- Contexts can be isolated with "vhosts"
- ACLs (vhosts & queues permissions) can be assigned to users
  - configure / read / write
  - can be applied with regex
- Useful tip for my past self
  - **Put ACLs from the beginning!**



New connections love TCP packets



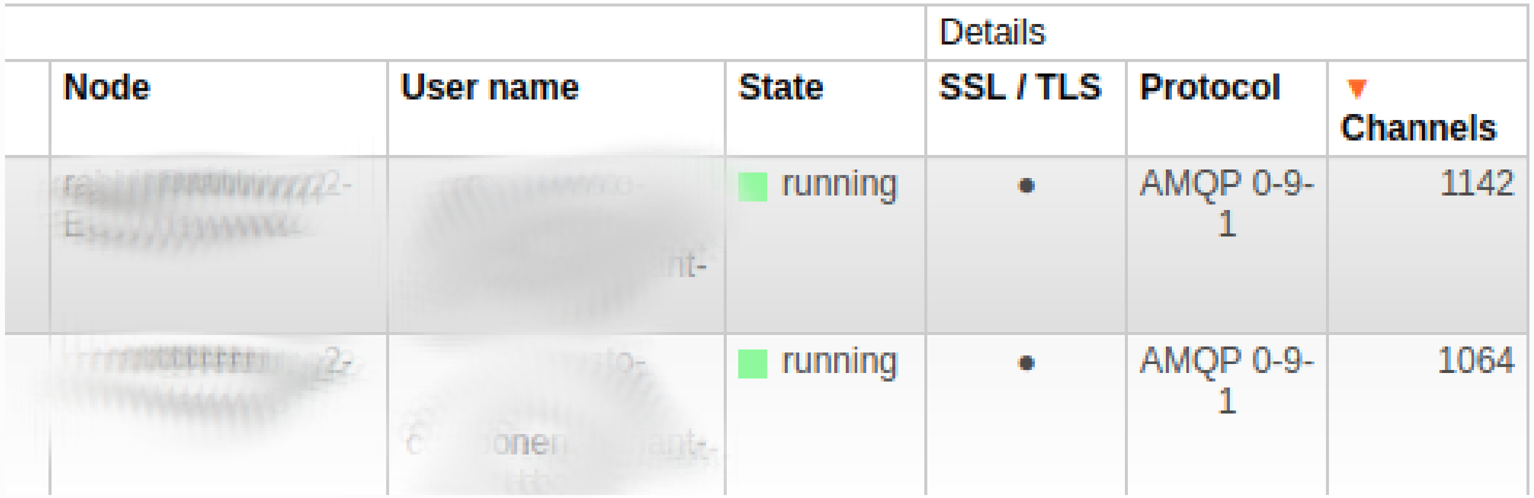
- AMQP connections: 7 TCP packets
- AMQP channel: 2 TCP packets
- AMQP publish: 1 TCP packets (more for larger messages)
- AMQP close channel: 2 TCP packets
- AMQP close connection: 2 TCP packets
- Total 14-19 packets (+ Acks)

- N°1 rule in Lovisa Johansson's *13 Common RabbitMQ Mistakes*
- Obviously: don't open a connection for each message 🤦

Connections / channels

A bit less obvious:

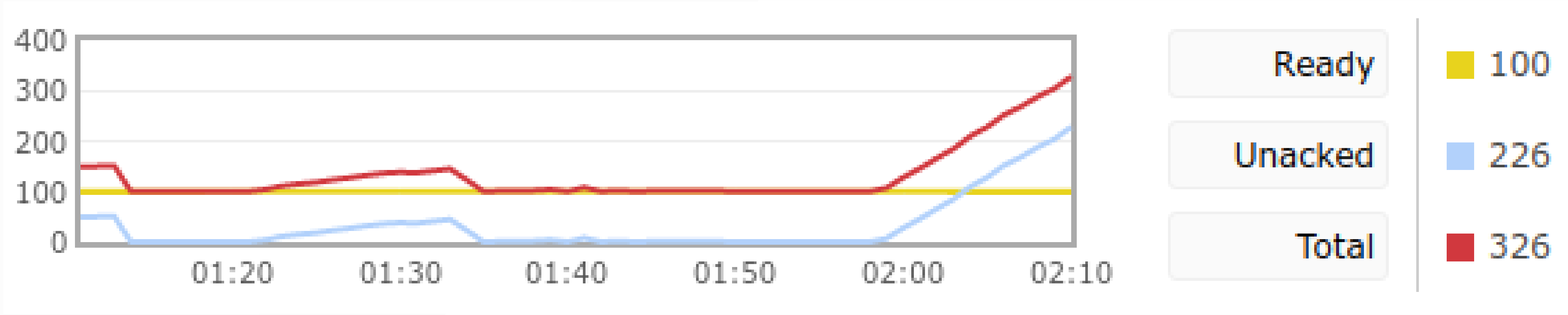
- Don't open a channel for each publish
- **Don't forget to close channels** 😱



				Details		
	Node	User name	State	SSL / TLS	Protocol	▾ Channels
	[redacted]	[redacted]	🟩 running	●	AMQP 0-9-1	1142
	[redacted]	[redacted]	🟩 running	●	AMQP 0-9-1	1064

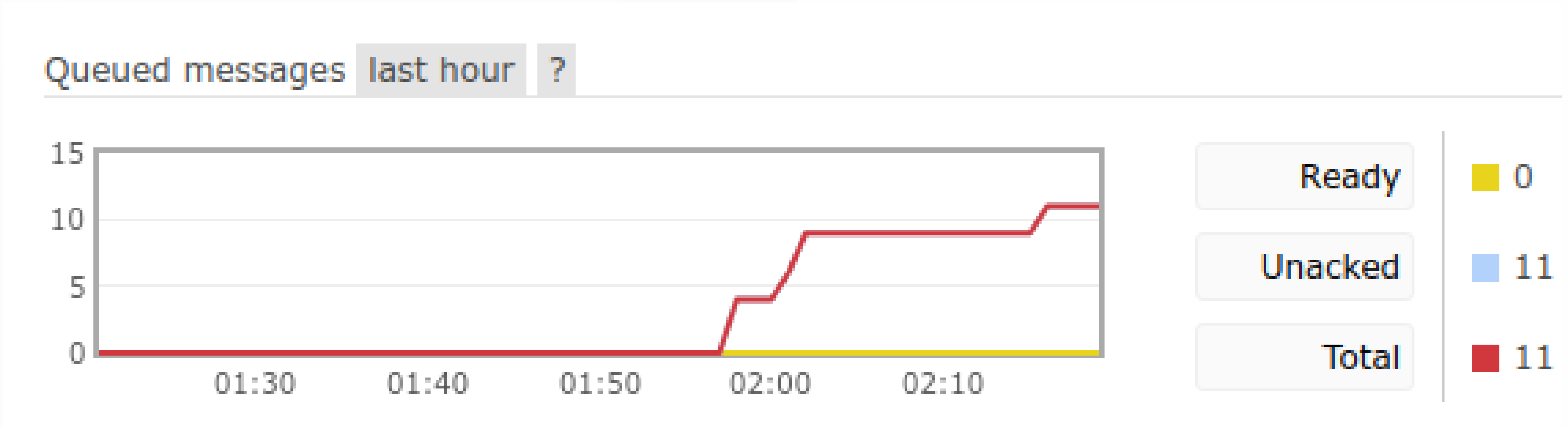
Prefetch

- Send messages to consumers even if **ack** hasn't yet been received
- Useful if you app can process messages really fast or in parallel
- More messages in the wild when something wrong happens 😈



Examples of issues with prefetch

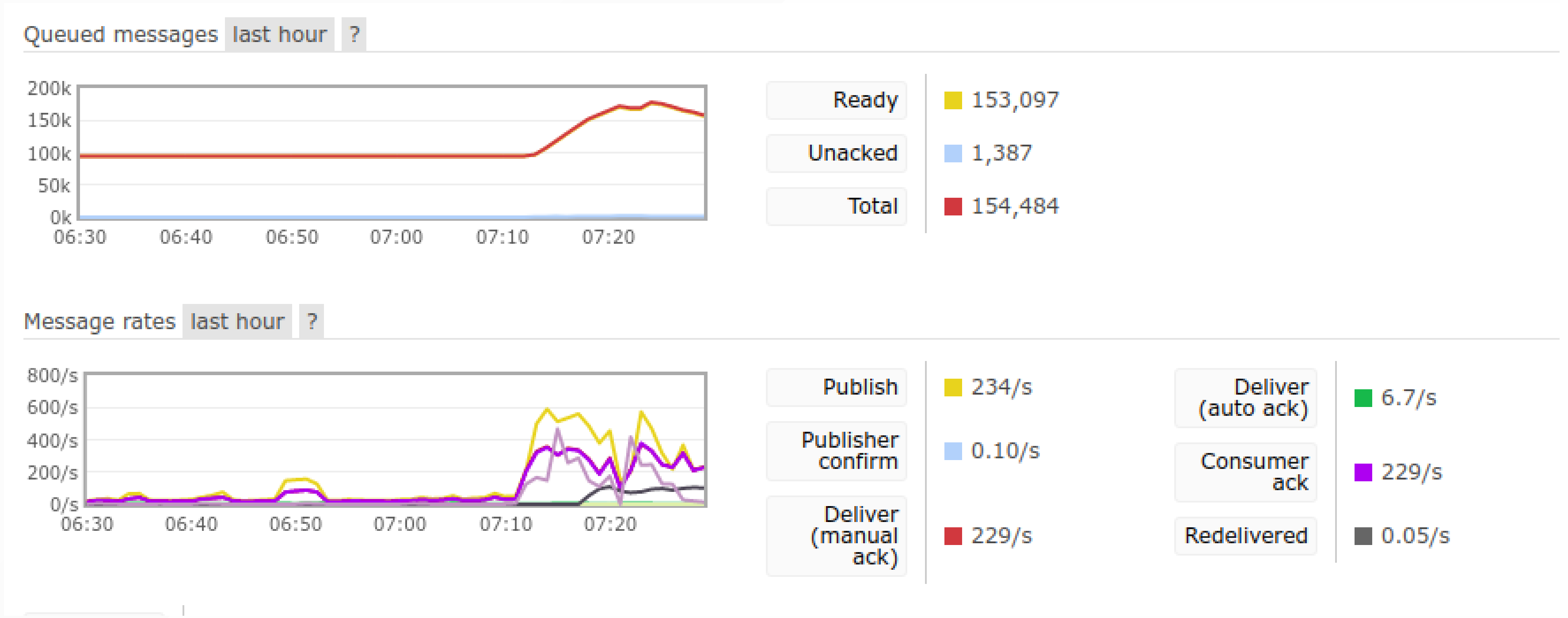
- Don't assume your app/framework correctly handles parallel treatment of messages
- Don't forget to catch errors (especially in threads)



Ready	🟨 0
Unacked	🟦 11
Total	🟥 11

Leaverage the power of TTLs and DLQs

- Keep your queues short and empty them quickly
  - If possible add *TTLs*, *queue length* and *message max size*
  - Be careful when reject + requeing



Ready	🟨 153,097
Unacked	🟦 1,387
Total	🟥 154,484

Publish	🟨 234/s	Deliver (auto ack)	🟩 6.7/s
Publisher confirm	🟦 0.10/s	Consumer ack	🟪 229/s
Deliver (manual ack)	🟥 229/s	Redelivered	⬛ 0.05/s

Observe

- RabbitMQ management plugin is cool
  - put it on more than one node
- Grafana dashboards are better
  - Use prometheus to scrape your cluster/queues
  - 3.8+: `rabbitmq-prometheus` plugin rather than external exporters

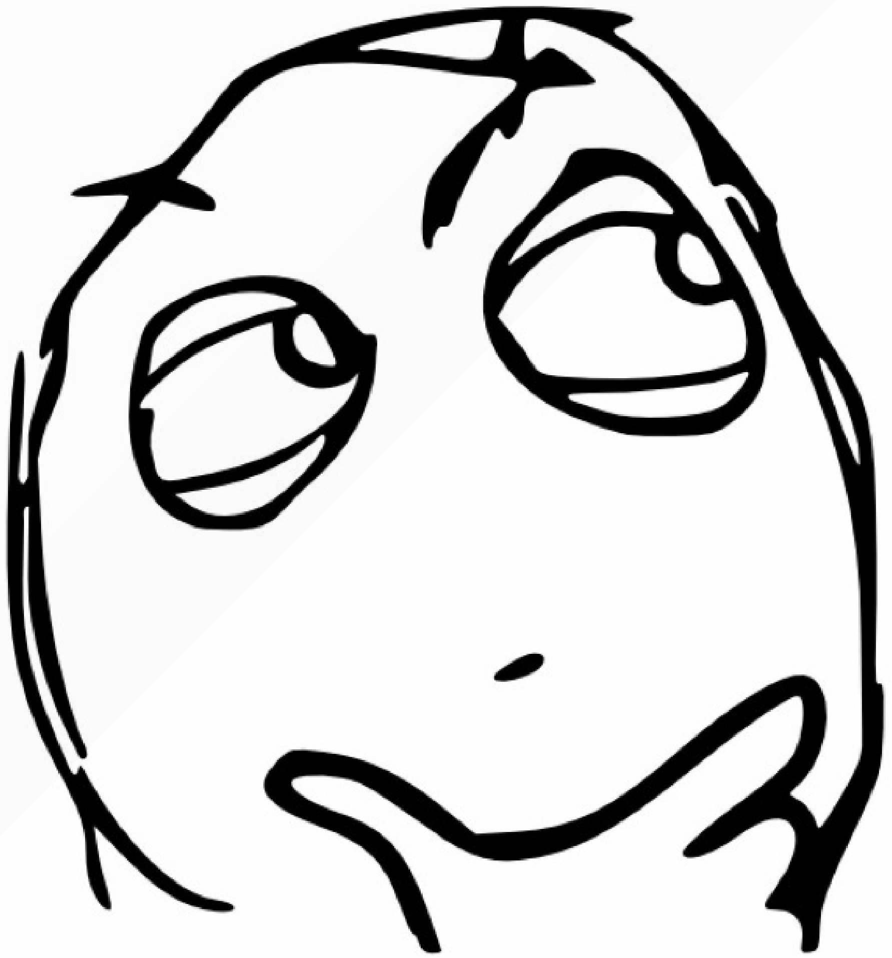
“Now, I know what to fix!”

Conclusion

- RabbitMQ clusters aren't what you think they are
    - What you want are quorum queues / HA queues
- RabbitMQ ain't Kafka
    - don't store 1M messages for a long time (except in streams 😉)
- Use connections / channels wisely
- BONUS:
    - Secure from the start (ACLs)
    - observe, observe, observe!!

That's all folks

Do you have any questions?



Bibliography

Best practices and advices

- CloudAMQP: 13 Common RabbitMQ Mistakes and How to Avoid Them
- RabbitMQ blog: Some queuing theory: throughput, latency and bandwidth
- RabbitMQ official documentation
- RabbitMQ official documentation: Tutorials
- Grafana dashboard for official prometheus exporter
- RabbitMQ: best practices collection
- RabbitMQ reliability guide