April 24th 2014 - GopherCon

Matt Reiferson - CTO at Torando Labs

NSQ

a realtime distributed messaging platform https://github.com/bitly/nsq @imsnakes





HI, I'M MATT



WHAT EVEN IS NSQ?

NOT STABLE QUEUE

NAMING SUCKS QUEUE

NEW SIMPLE QUEUE



SPRAY SOME NSQ ON IT







BIGTHINGS HAVE SMALL BEGINNINGS

•boss says: "I want metrics!"

- •single host
- •synchronous writes





BIGTHINGS HAVE SMALL BEGINNINGS

•boss says: "I want metrics!"

- •single host
- synchronous writes





... AND THEN THEY EATYOU





... AND THEN THEY EATYOU ********** nightmare



BUTTHERE'S HOPE





provide a unifying distributed system to receive and disseminate event data

MESSAGING PATTERNS

Producer

























Producer





ml ConsumerB

Producer



PS

m2 ConsumerB

Producer

horizontal scalability





Q



























Producer







Producer

horizontal scalability









FAILURE





fault tolerance







FAILURE





fault tolerance







FAILURE











FAILURE











FAILURE











FAILURE









FAILURE









FAILURE



Q





FAILURE





EVEN MORE FAILURE

Q

Producer





welp

EVEN MORE FAILURE





welp




















































NSQD

- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



combine pubsub, distribution, and queueing

- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



nsqd

- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- a **topic** is a distinct stream of messages
- a topic has one or more channels
- topics and channels are created at runtime
- messages are **pushed** to consumers



- topics and channels are independent
- configurable high water mark (disk persistence)
- "bounded" memory footprint

```
for msg := range c.incomingMsgChan {-
select {¬
case c.memoryMsgChan <- msg:¬</pre>
default:-
    err := WriteMessageToBackend(&msgBuf, msg, c)
    if err != nil {¬
        // log whatever-
```





THERE AND BACK AGAIN

DE-COUPLE

• PUB locally to **nsqd** via HTTP

•perform work async

•co-locate everything (silo)



SCALE HORIZONTALLY



1 I I I

metrics

. . .

A MESSAGE QUEUE IS BORING (IN ISOLATION)

NSQLOOKUPD

no centralized broker

 nsqlookupd instances are independent (no coordinatation)





TYPICAL NSQ CLUSTER











no centralized broker

 nsqlookupd instances are independent (no coordinatation)



TYPICAL NSQ CLUSTER





PUBLISH









no centralized broker

 nsqlookupd instances are independent (no coordinatation)





no centralized broker

 nsqlookupd instances are independent (no coordinatation)



no centralized broker

 nsqlookupd instances are independent (no coordinatation)



SUBSCRIBE
... AND AGAIN

- introduce nsqlookupd
- producers and consumers
 come and go
- other services can discover and subscribe to this topic



- introduce nsqlookupd
- producers and consumers
 come and go
- other services can discover and subscribe to this topic



- introduce nsqlookupd
- producers and consumers
 come and go
- other services can discover and subscribe to this topic



- introduce nsqlookupd
- producers and consumers
 come and go
- other services can discover and subscribe to this topic



- introduce nsqlookupd
- producers and consumers
 come and go
- other services can discover and subscribe to this topic





TO INFINITY AND BEYOND



TO INFINITY AND BEYOND



Datacenter B















OPS

GUARANTEES

•messages are delivered at least once •messages are not durable (by default) messages received are un-ordered

consumers eventually find all topic producers

- processing fails?
 - slow down rate
- bad message?
 - •limit attempts
 - delay re-processing



- processing fails?
 - slow down rate
- bad message?
 - •limit attempts
 - delay re-processing



- processing fails?
 - slow down rate
- bad message?
 - •limit attempts
 - delay re-processing





- processing fails?
 - slow down rate
- bad message?
 - •limit attempts
 - delay re-processing





- processing fails?
 - slow down rate
- bad message?
 - •limit attempts
 - delay re-processing





- processing fails?
 - slow down rate
- bad message?
 - •limit attempts
 - delay re-processing






































TOPIC & CHANNEL PAUSING



TOPIC & CHANNEL PAUSING



ONE MORETHING

- •#ephemeral channels
- channel sampling
- •TLS / Snappy
- telemetry over statsd / HTTP

O O NSQ correlated_decodes ×				
C f https://nsqadmin-ec2.bitly.net/topic/correlated_decodes				
NSQ Streams Nodes Counter I 48h -	NSQ on github	v0.2.16-alpha		

Streams / correlated_decodes

Topic: correlated_decodes

Delete Topic

Topic Message Queue

nsqd Host	Depth	Memory + Disk	Messages	Channels
cordreader04.ec2.bitly.net:4151	where where where o	0 + 0	952,448,957	3
cordreader05.ec2.bitly.net:4151	Who would a work o	0 + 0	956,705,323	3
Total:	When my Manual O	0 + 0	1,909,154,280	3

Channel Message Queues

Channel	Depth	Memory + Disk	In-Flight	Deferred	Requeued	Timed Out	Messages	Connections
nsq_to_file	manun 0	0 + 0	1,007	0	123	123	1,909,154,280	6
privacy_store	where where the other of the other o	0 + 0	40	0	22	22	1,909,154,281	16
stageprivacystore01	Manual O	0 + 0	40	0	0	0	1,909,154,281	4

SUMMARY

- •github.com/bitly/nsq
- 10,300 lines of Go
- 19 client libraries, 11 languages

• over 2 years in production





SUMMARY

- •github.com/bitly/nsq
- 10,300 lines of Go
- 19 client libraries, 11 languages

• over 2 years in production









DRAMA FEVER



REONOMY



Path



simplereach

IN PRODUCTION

EnergyHub

Trendrr[.]





eventful







shoutout to **@jehiah** (co-author of NSQ)

Thanks!

@imsnakes

https://github.com/bitly/nsq