GOLANG AT BITLY

with a focus on **NSQ**

@mrwoofster

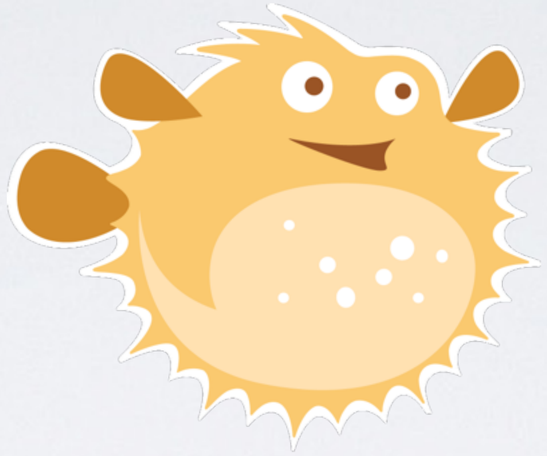
Michael Richman - App Engineer @ Bitly

GopherCon 2015

bitly

MICHAEL RICHMAN, BITLY DENVER
@mrwoofster

bitly



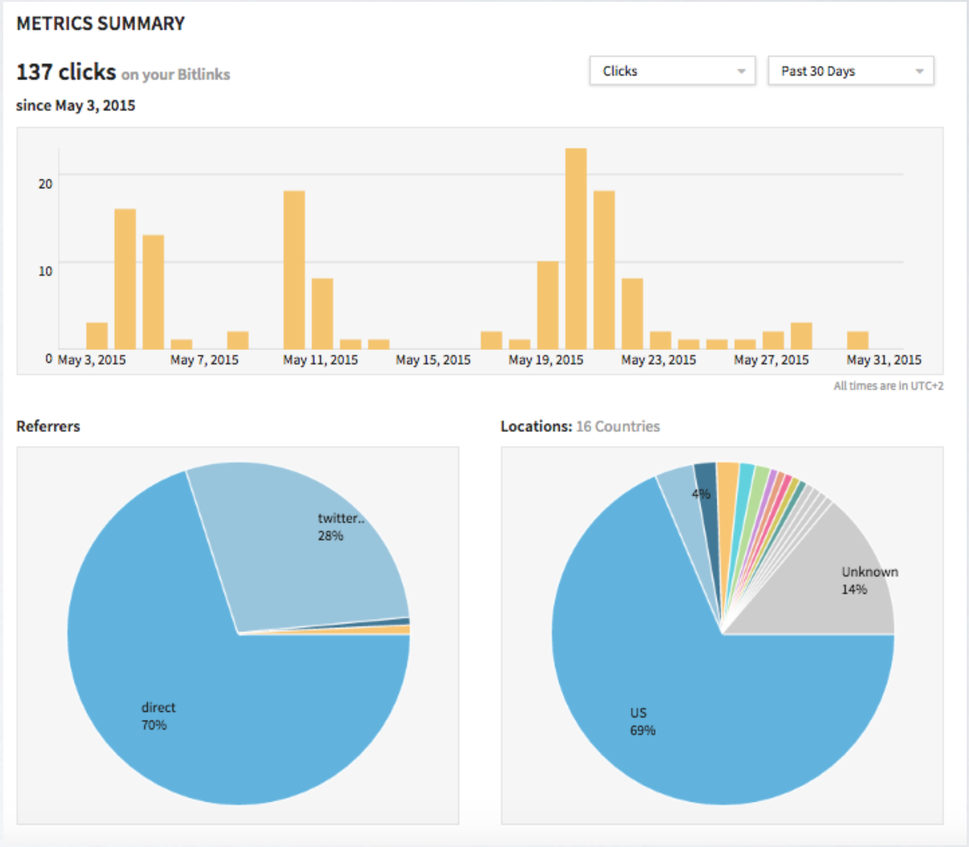
MICHAEL RICHMAN, BITLY DENVER

@mrwoofster

bitly

WHAT IS **BITLY**?

- Popular URL shortener (eg. bit.ly/AGb56v)
- analytics around **how** and **where** those links were shared
- **10 billion** clicks per month
- **8000** requests per second



bitly

GOLANG @ BITLY

- **NSQ** - realtime messaging platform written in Go
- **Queuereaders** to process messages
- **Core redirect service** (hybrid)
- **gomrjob** - Go library for Hadoop map reduce jobs

bitly

WHAT IS **NSQ**?

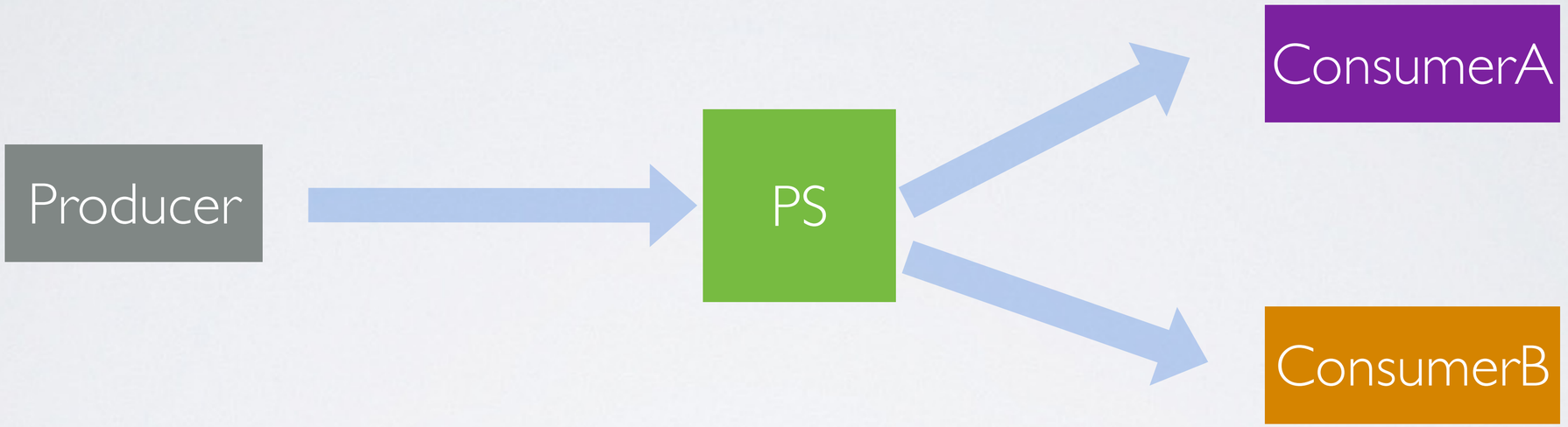
- Realtime distributed messaging platform
- Open-sourced by Bitly, at version: 0.3.5
- Written in Go
- 19 client libraries, 11 languages
- > 3 years in production
- nsq.io

bitly

MESSAGING PATTERNS

bitly

BROADCAST

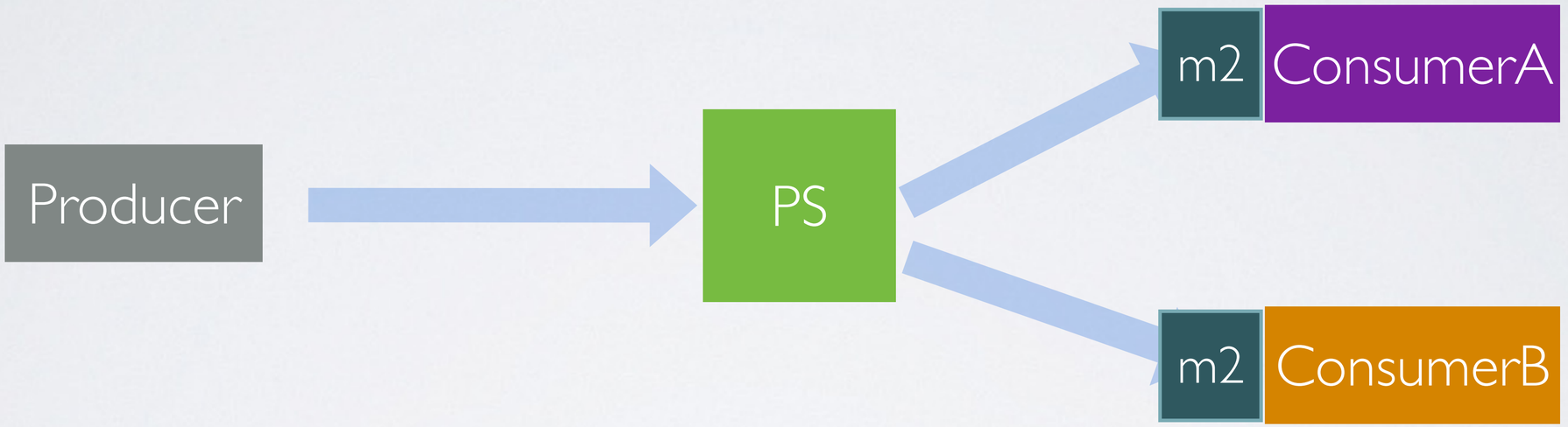


de-coupling of producers and consumers

multicast: message copied and delivered to n consumers

bitly

BROADCAST

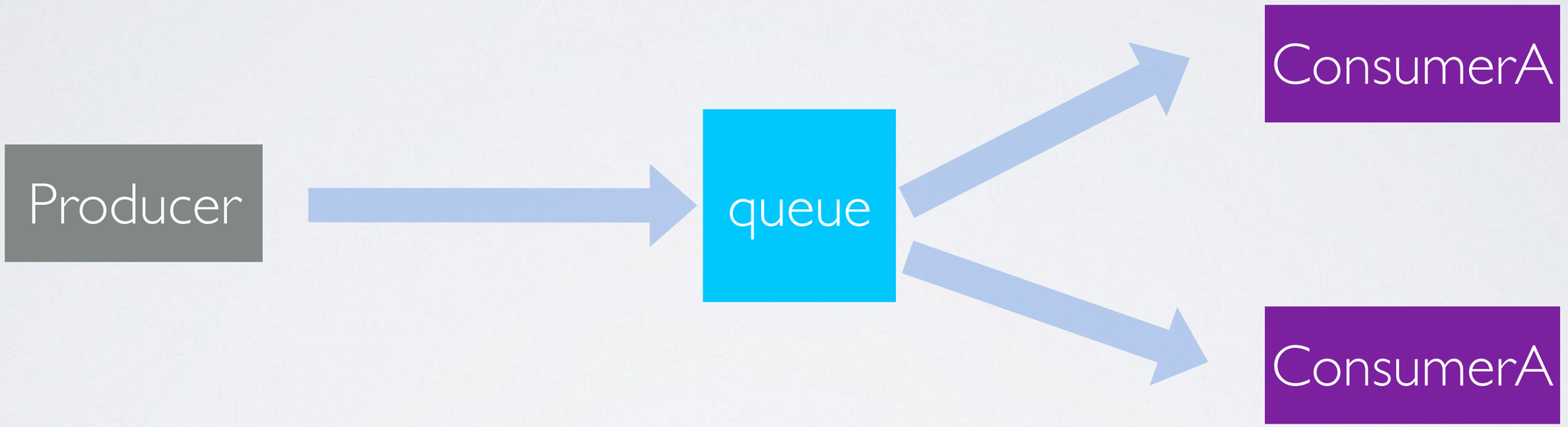


de-coupling of producers and consumers

multicast: message copied and delivered to n consumers

bitly

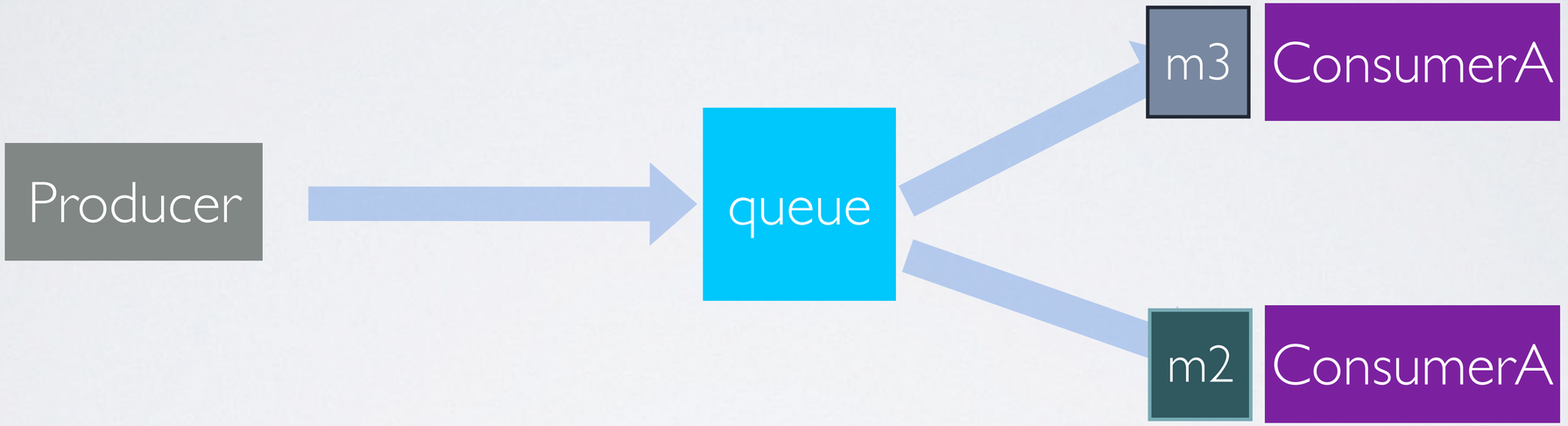
DISTRIBUTION



horizontal scalability

bitly

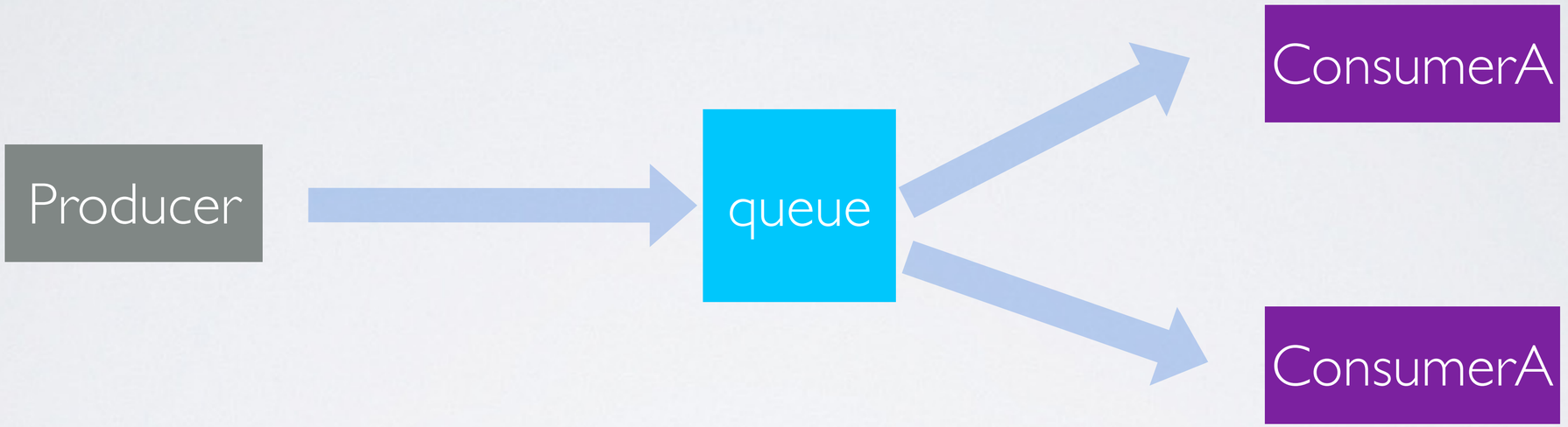
DISTRIBUTION



horizontal scalability

bitly

FAILURE



fault tolerance

bitly

FAILURE

fault tolerance

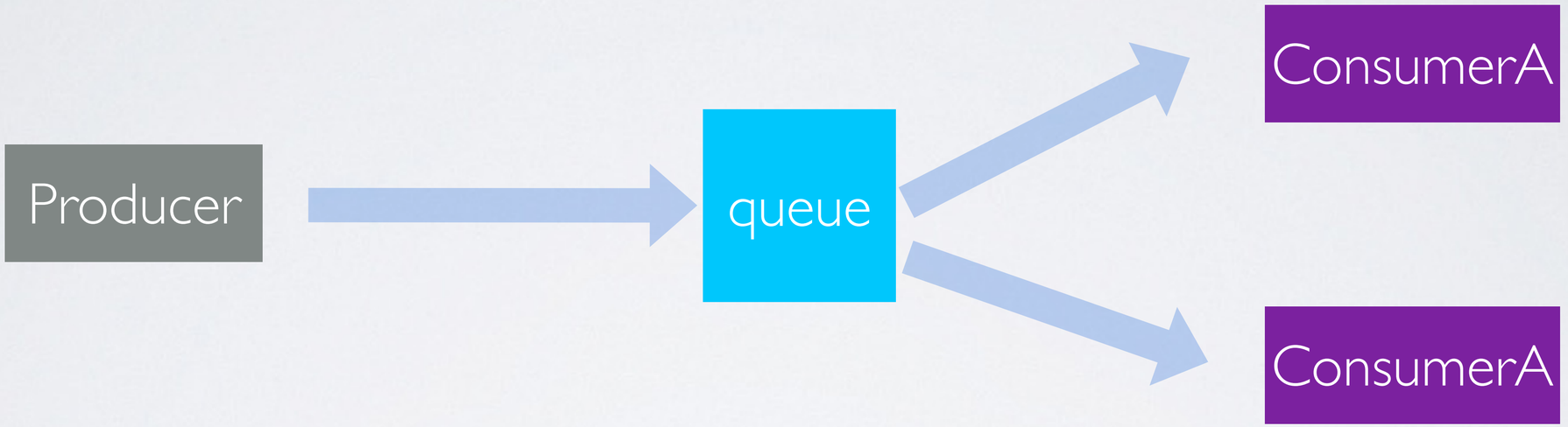
bitly

FAILURE

fault tolerance

bitly

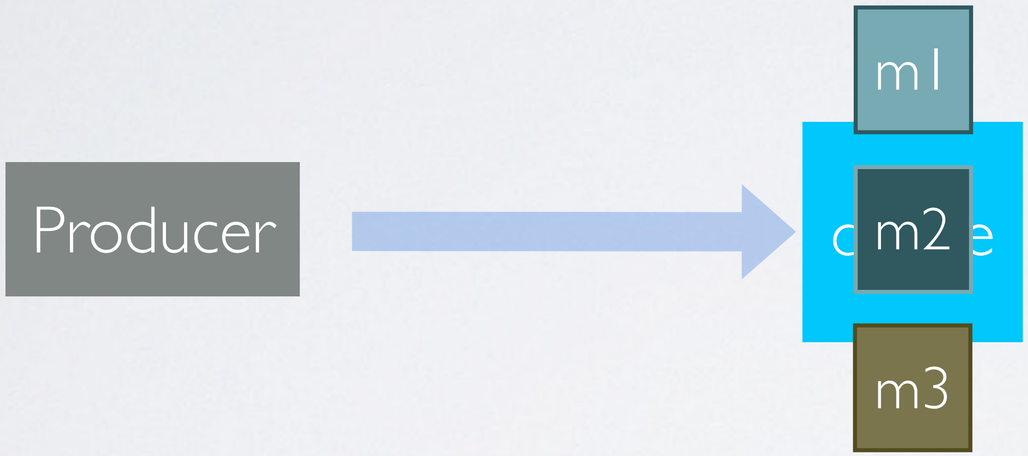
EVEN MORE FAILURE



when *all the things* fail

bitly

EVEN MORE FAILURE



when *all the things* fail

bitly

EVEN MORE FAILURE

when *all the things* fail

bitly

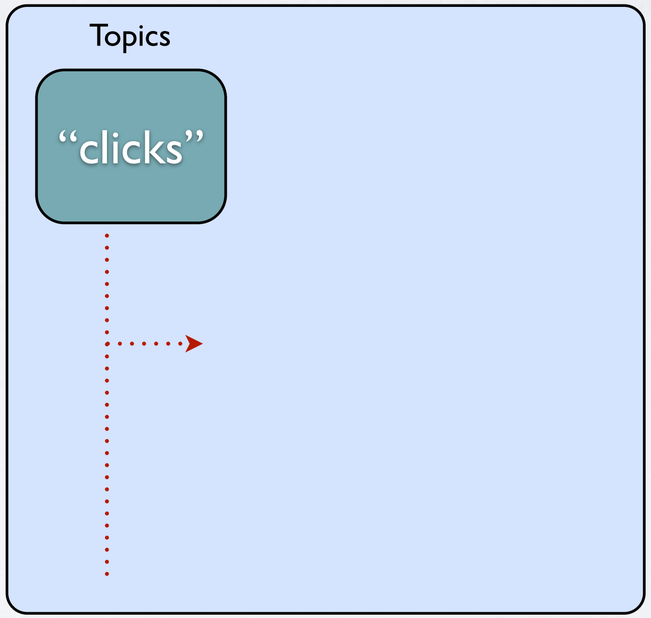
NSQD

bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

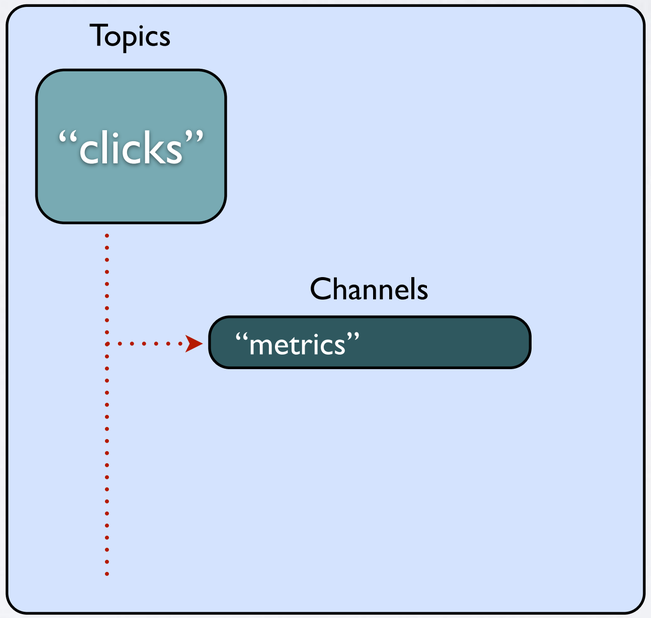


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

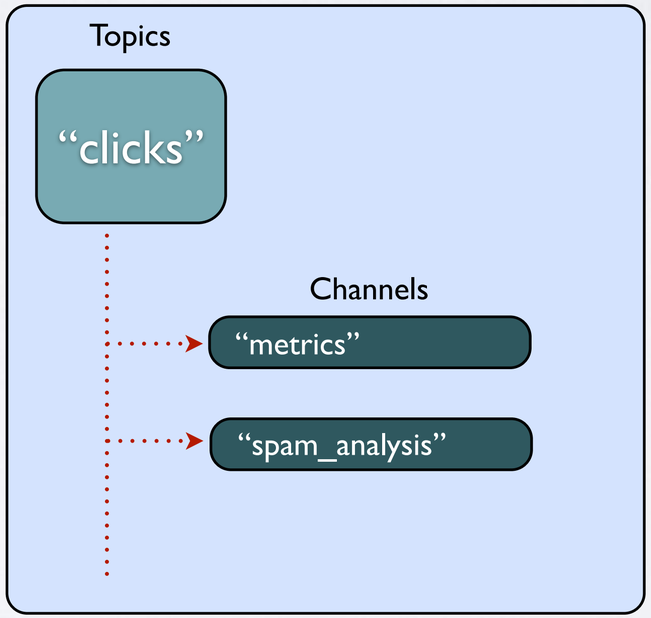


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

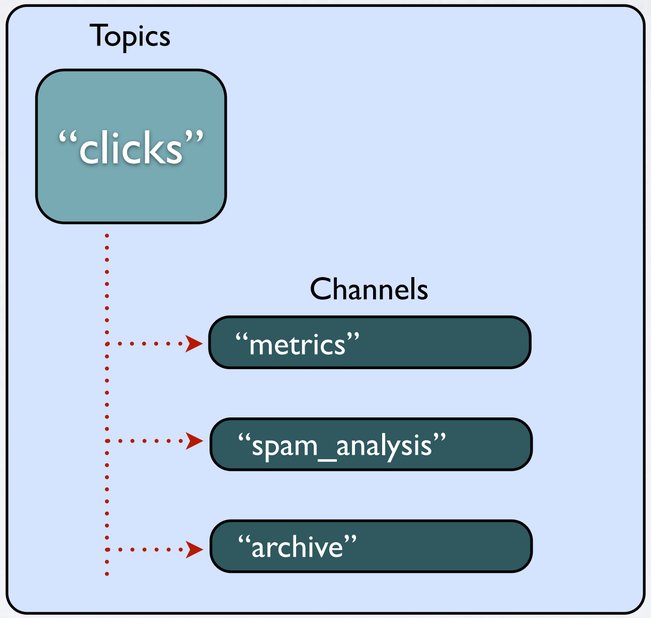


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

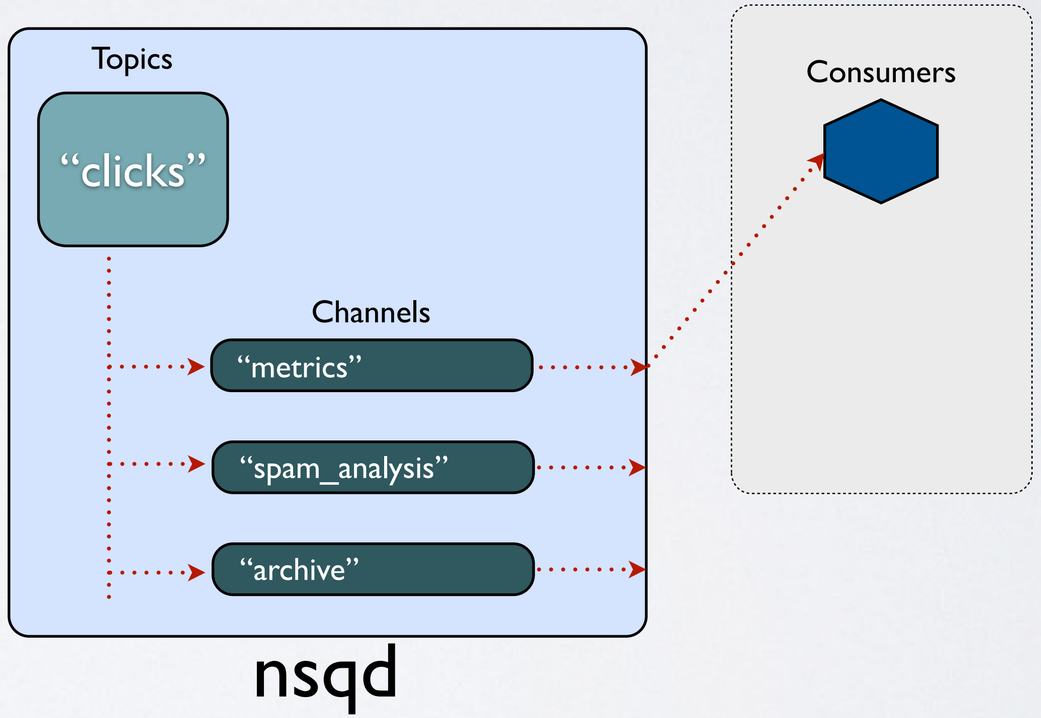


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

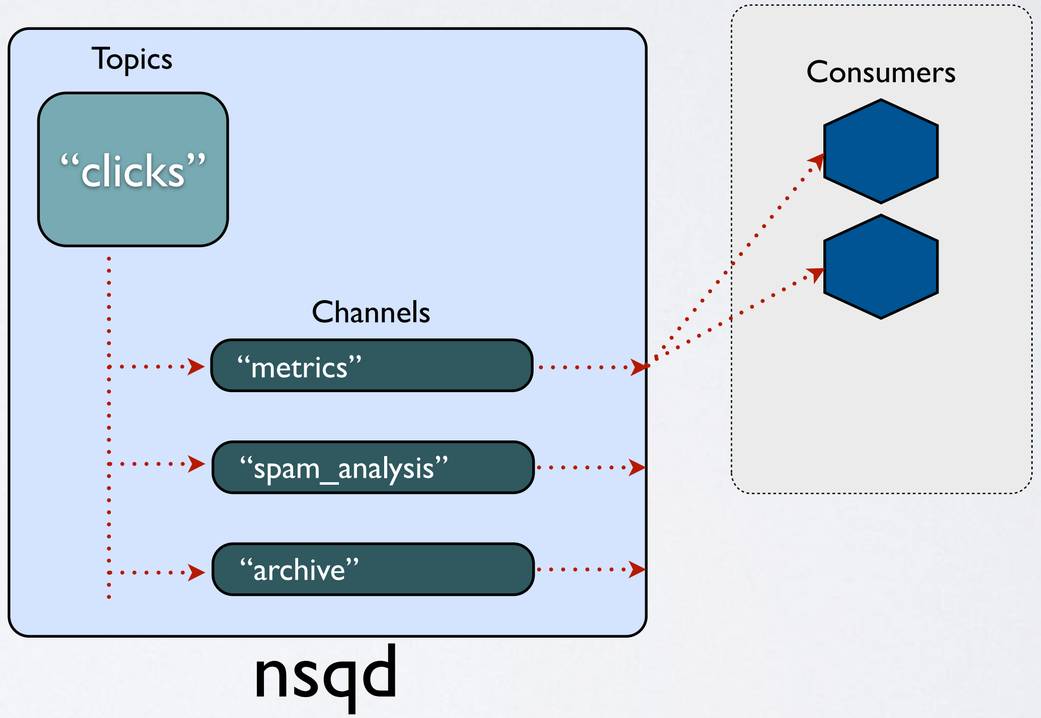


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

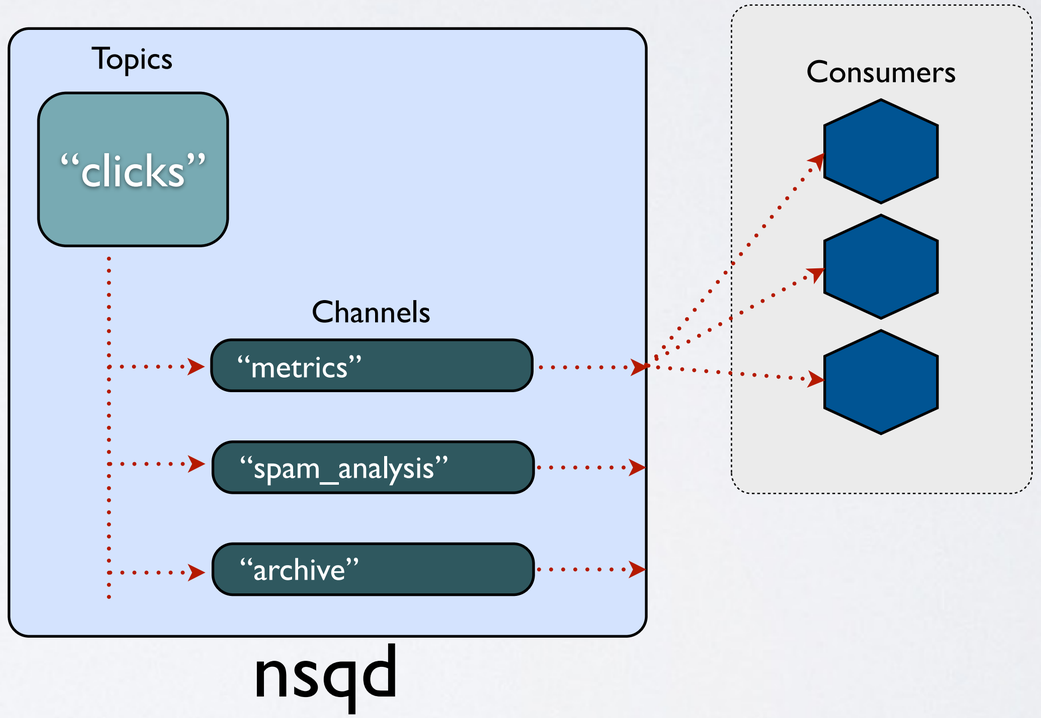


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

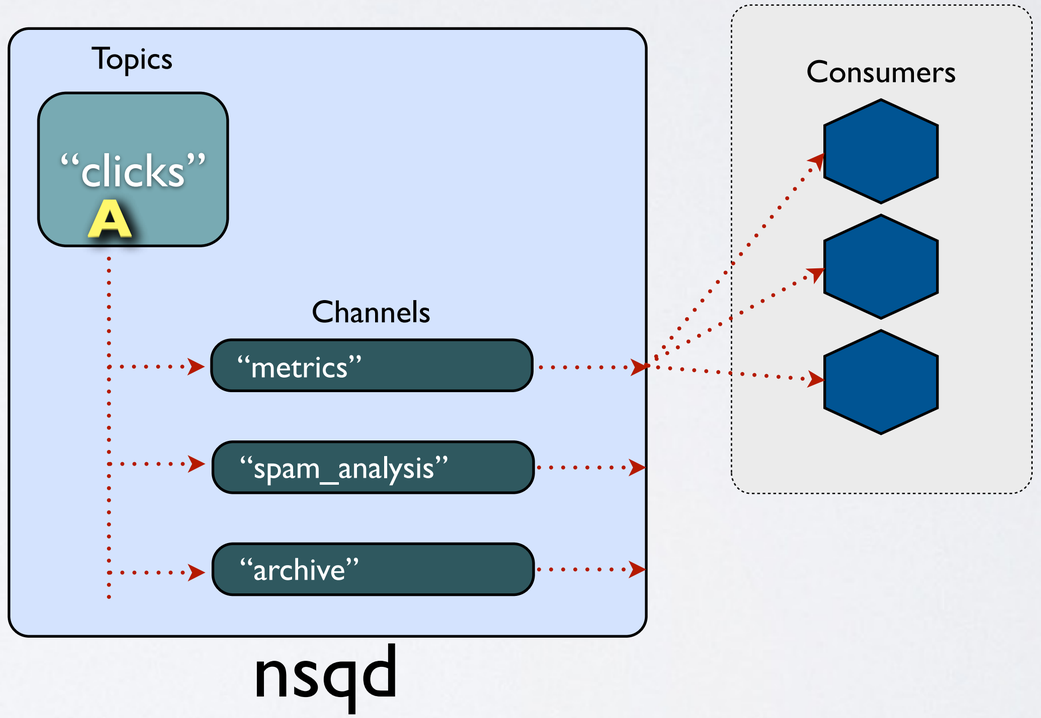


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

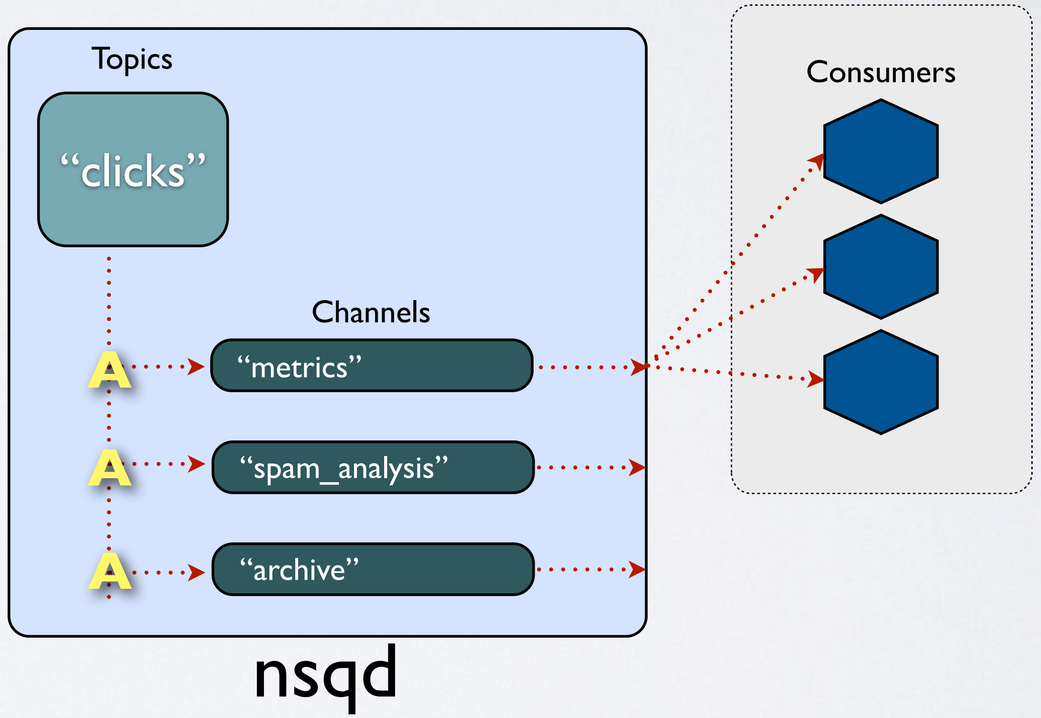


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

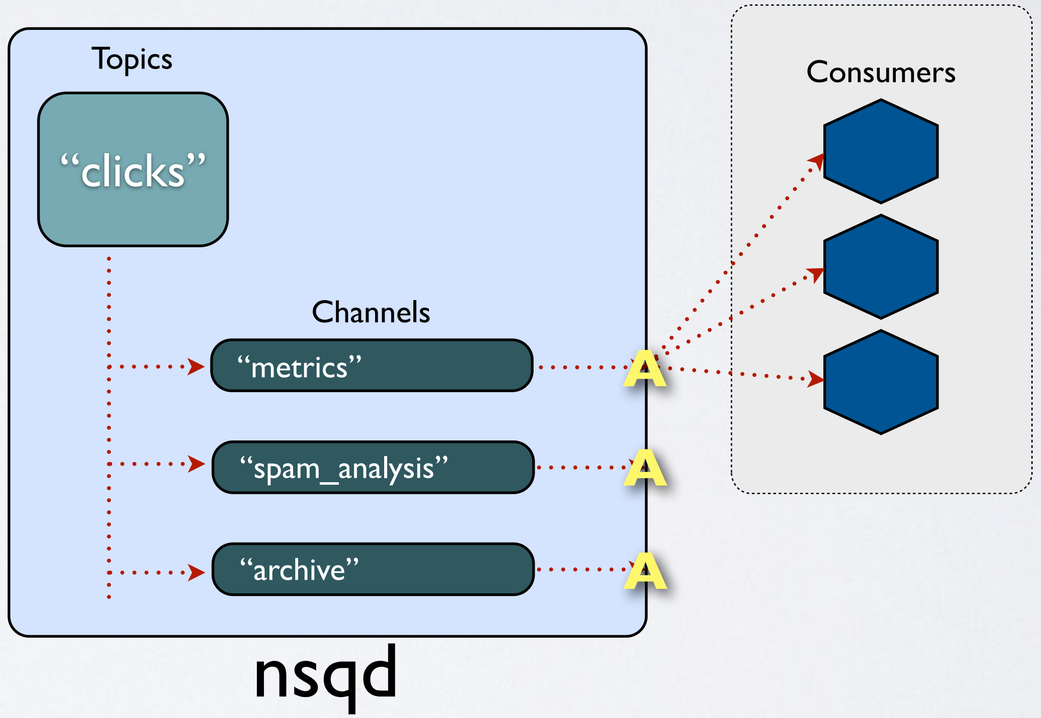


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

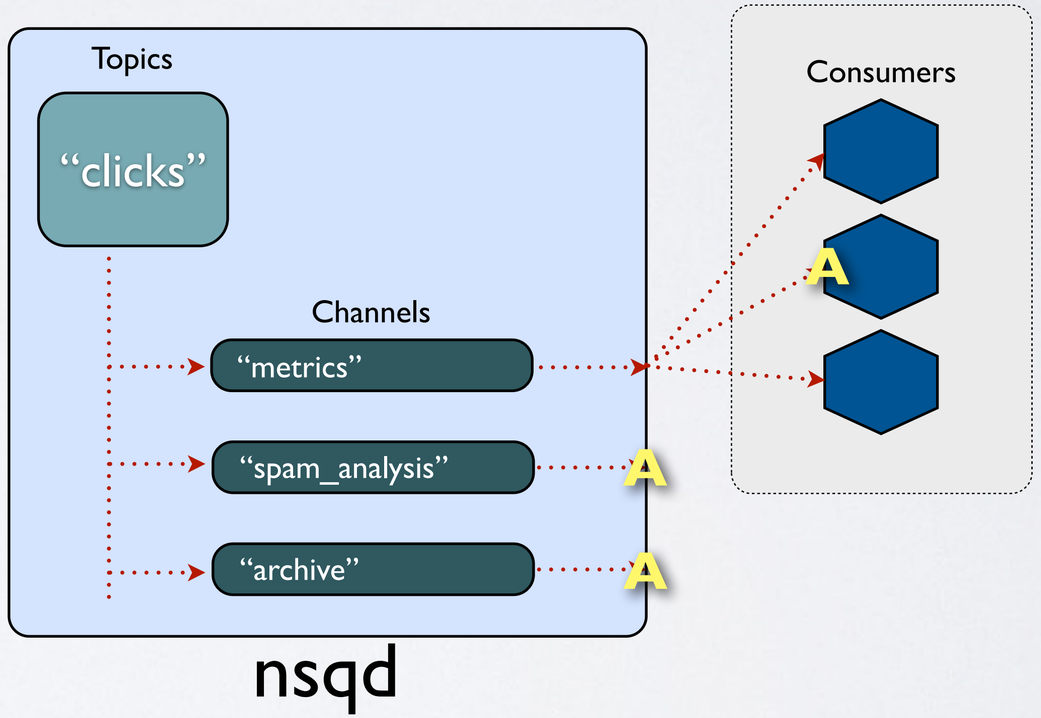


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

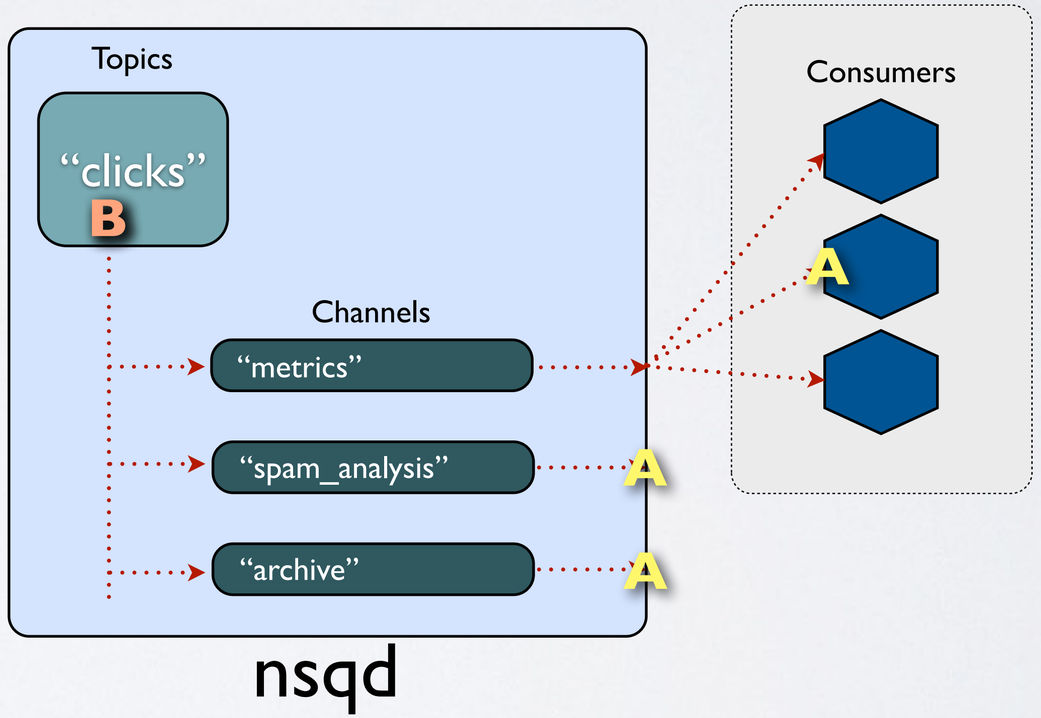


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

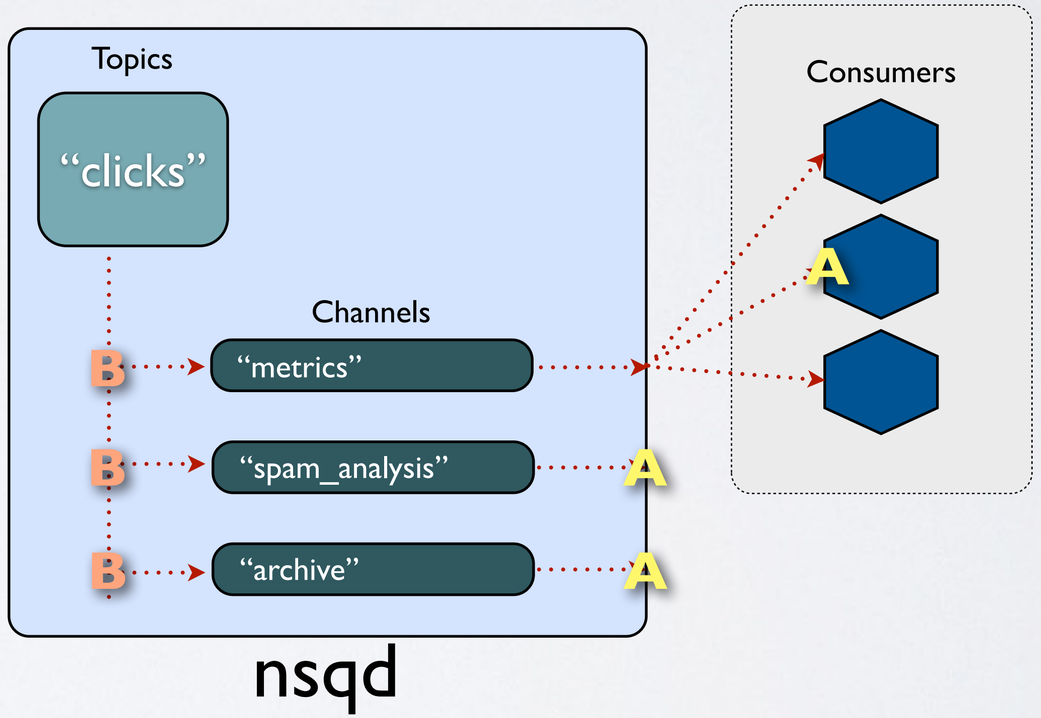


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

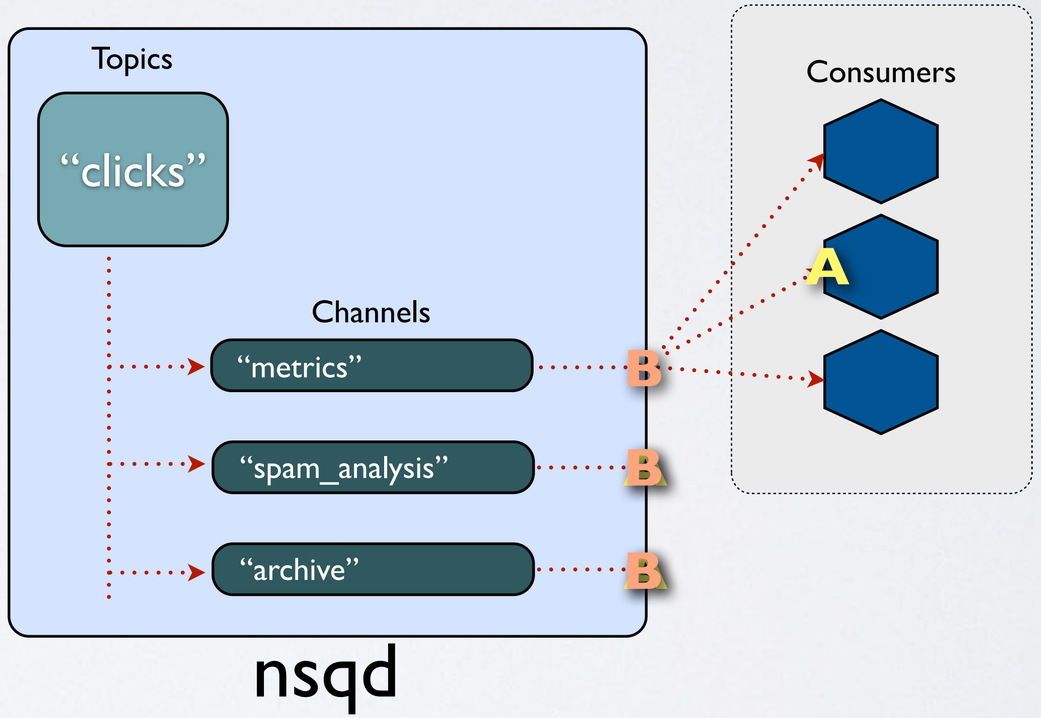


bitly

TOPICS AND CHANNELS

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 subscribing to a topic via a channel

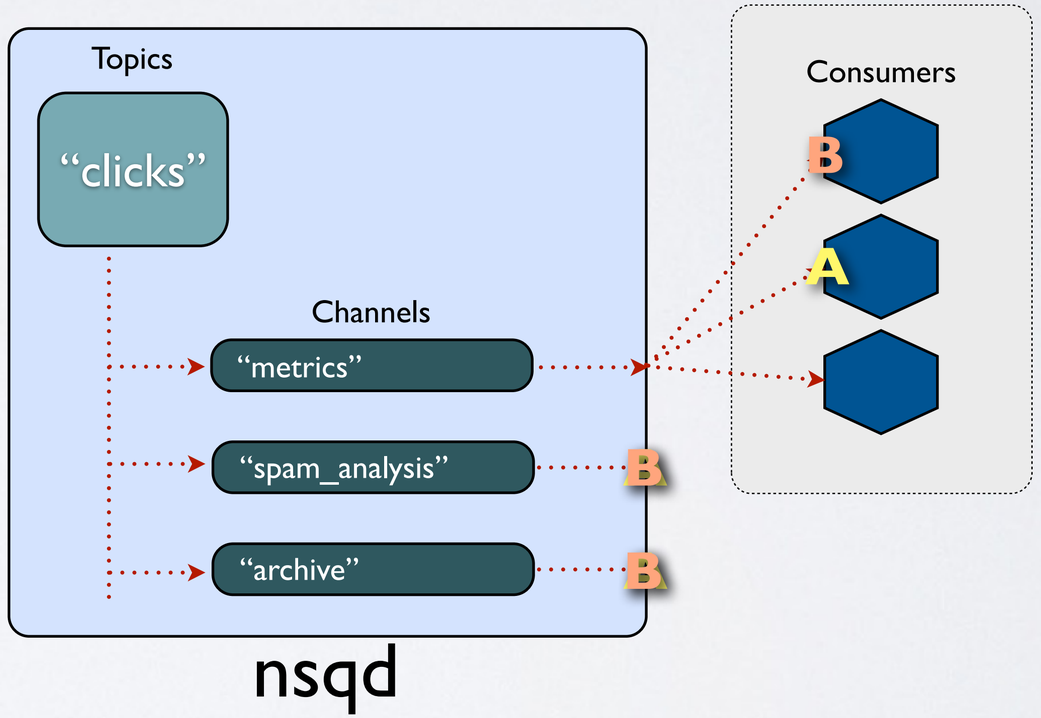


bitly

TOPICS AND CHANNELS

combine pubsub, distribution, and queueing

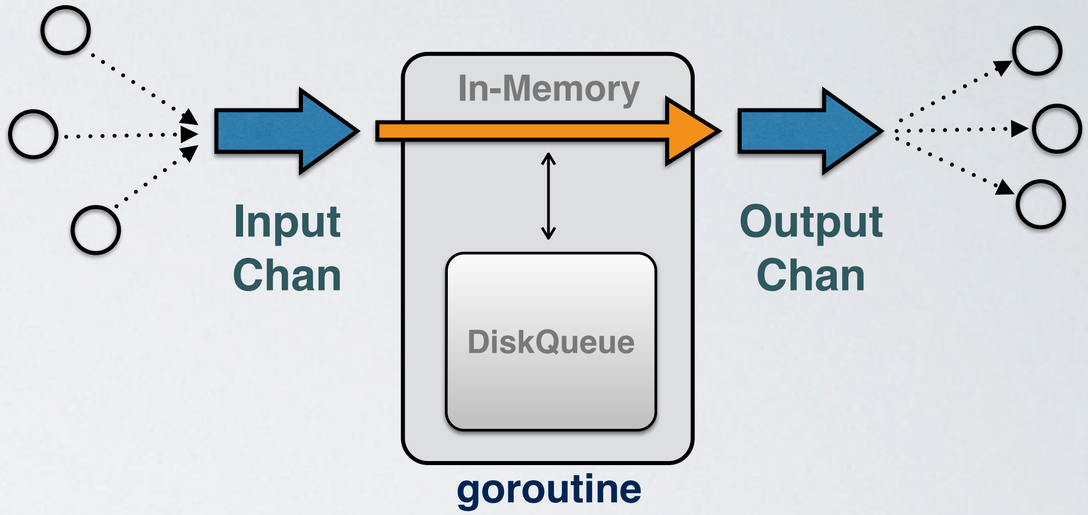
- a **topic** is a distinct stream of messages
- a **topic** has one or more **channels**
- topics and channels are created at **runtime**
- messages are **pushed** to consumers subscribing to a topic via a channel

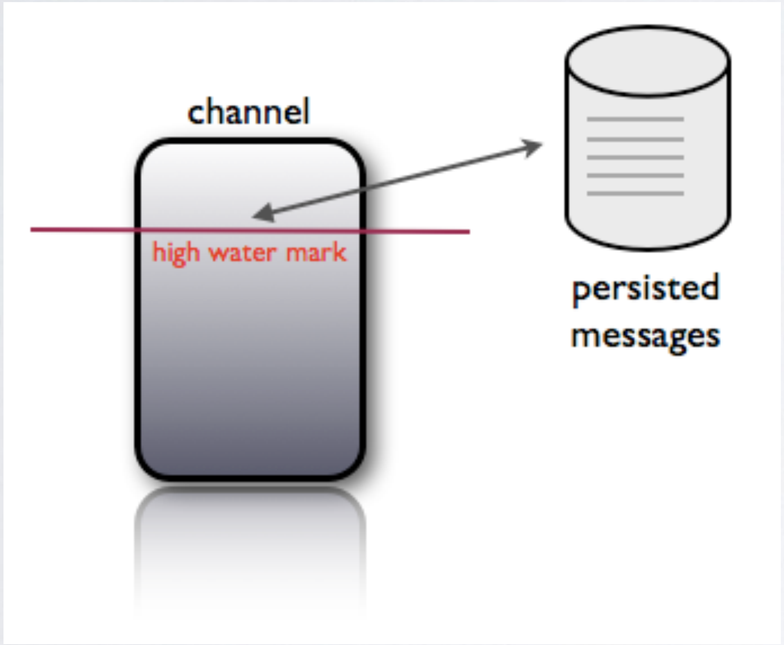


bitly

UNDER THE HOOD

- **topics** and **channels** are *independent*
- configurable high water mark (disk persistence) with --mem-queue-size

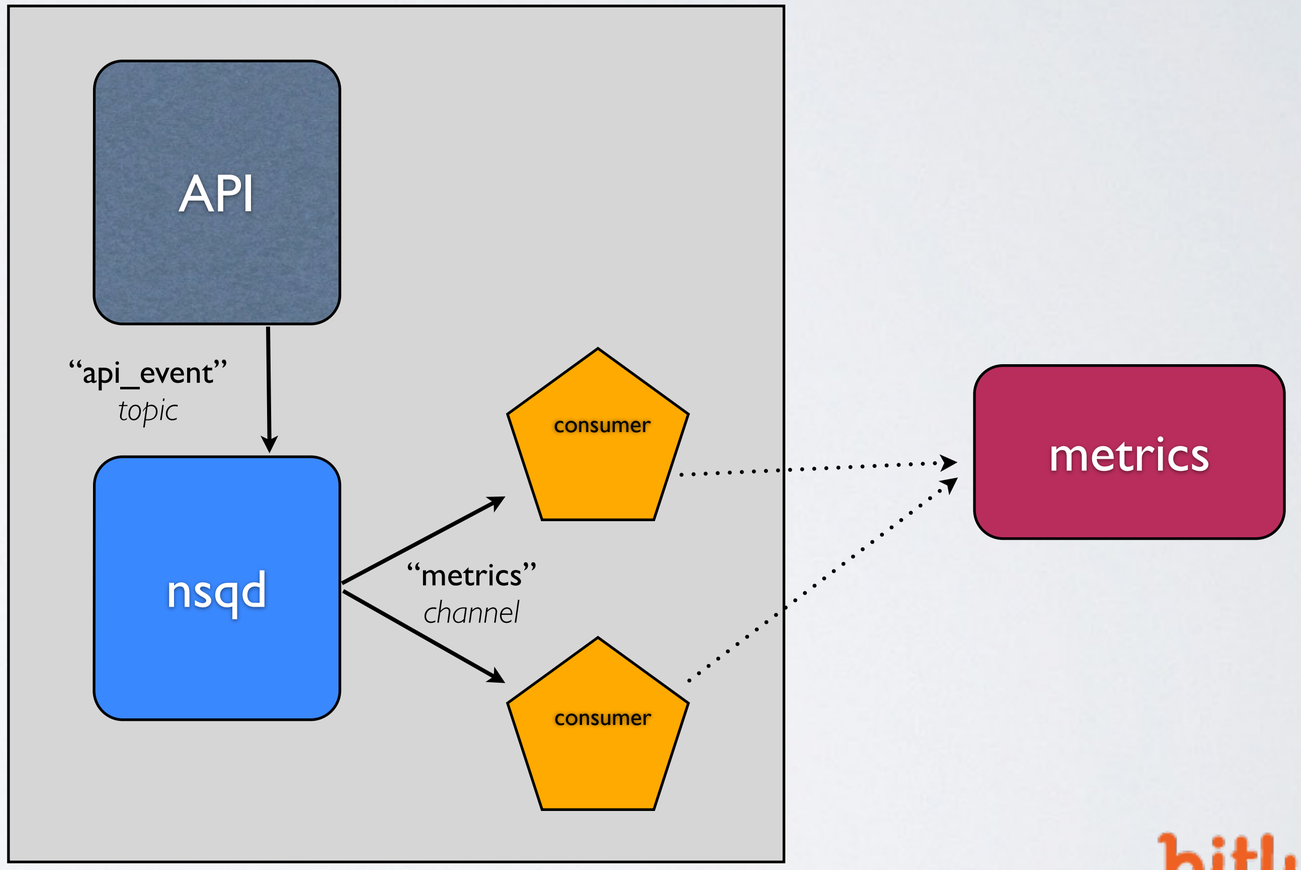





```
for msg := range c.incomingMsgChan {
    select {
    case c.memoryMsgChan <- msg:
    default:
        err := WriteMessageToBackend(&msgBuf, msg, c)
        if err != nil {
            // log whatever
        }
    }
}
```

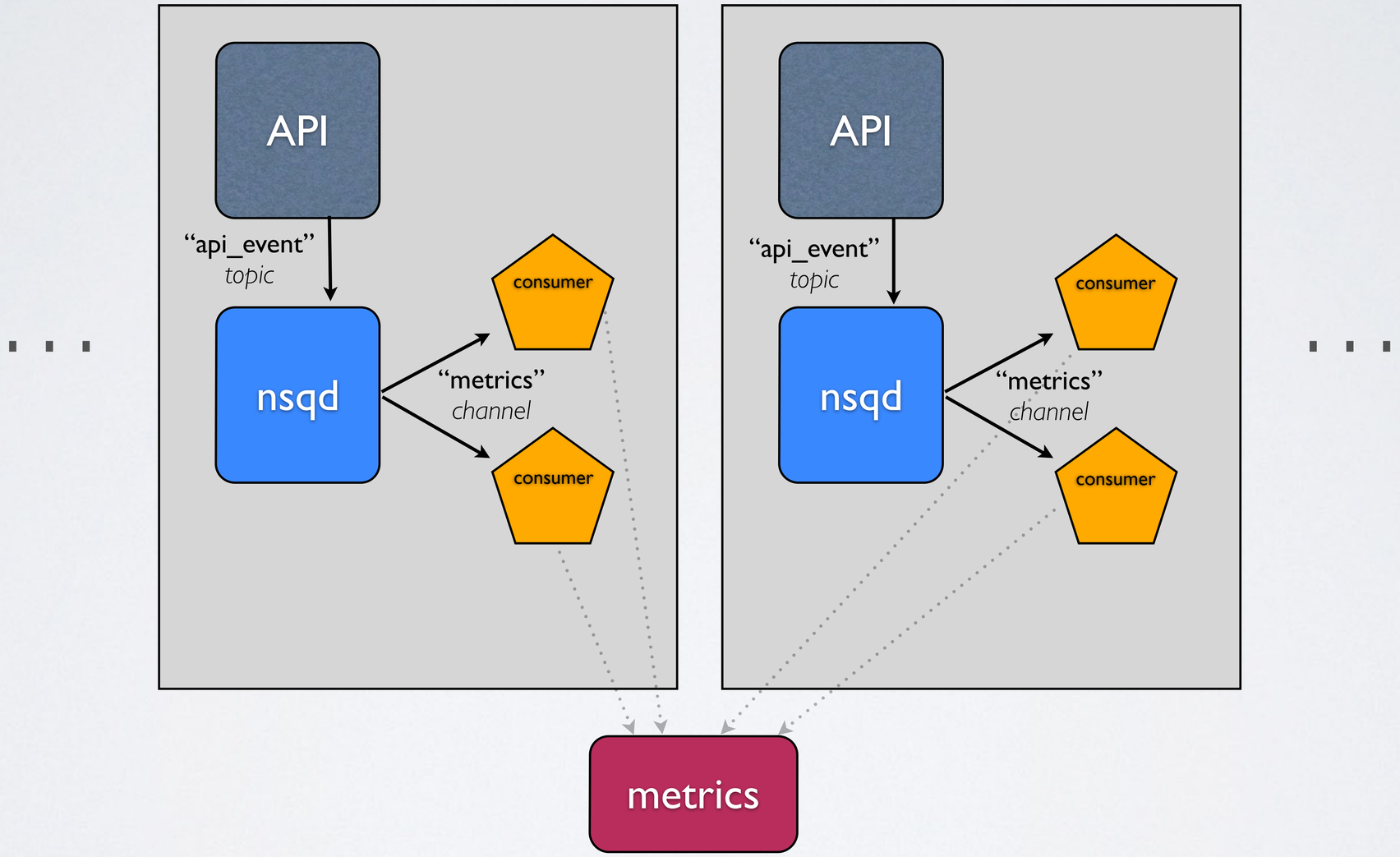
A SIMPLE EXAMPLE + NSQD

- introduce **nsqd**
- **de-coupled** production and consumption of data
- PUB locally to **nsqd** via HTTP
- perform work **async**
- co-locate everything (**silo**)



bitly

SCALE HORIZONTALLY

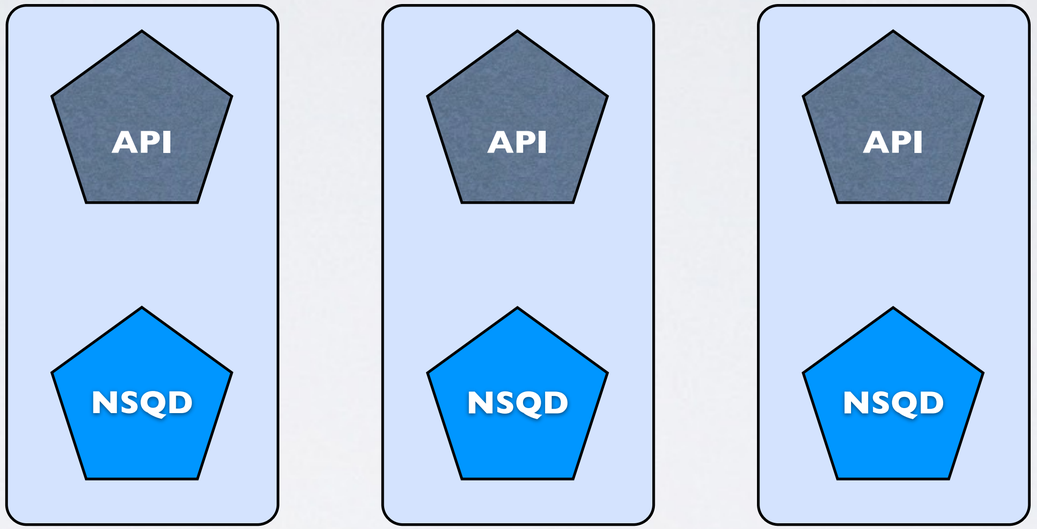


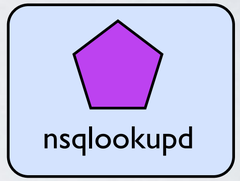
NSQLOOKUPD

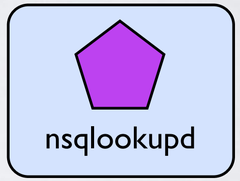
bitly

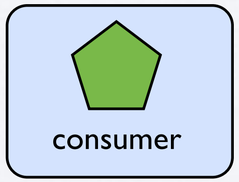
TYPICAL NSQ CLUSTER

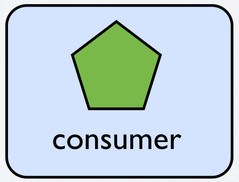
- enable *distributed* and *decentralized* topologies
- no centralized broker
- **nsqlookupd** instances are *independent* (**no coordinatation**)







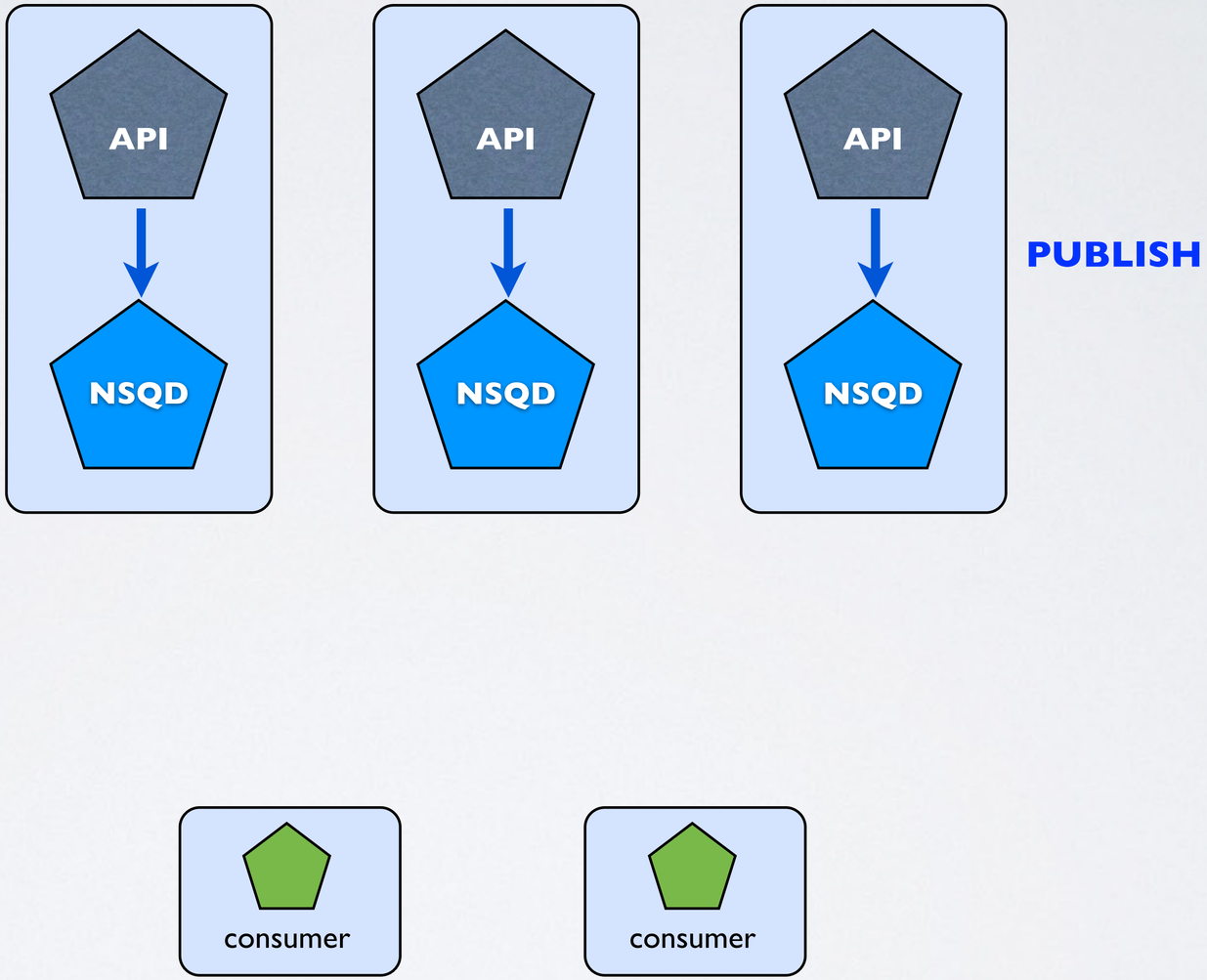




bitly

TYPICAL NSQ CLUSTER

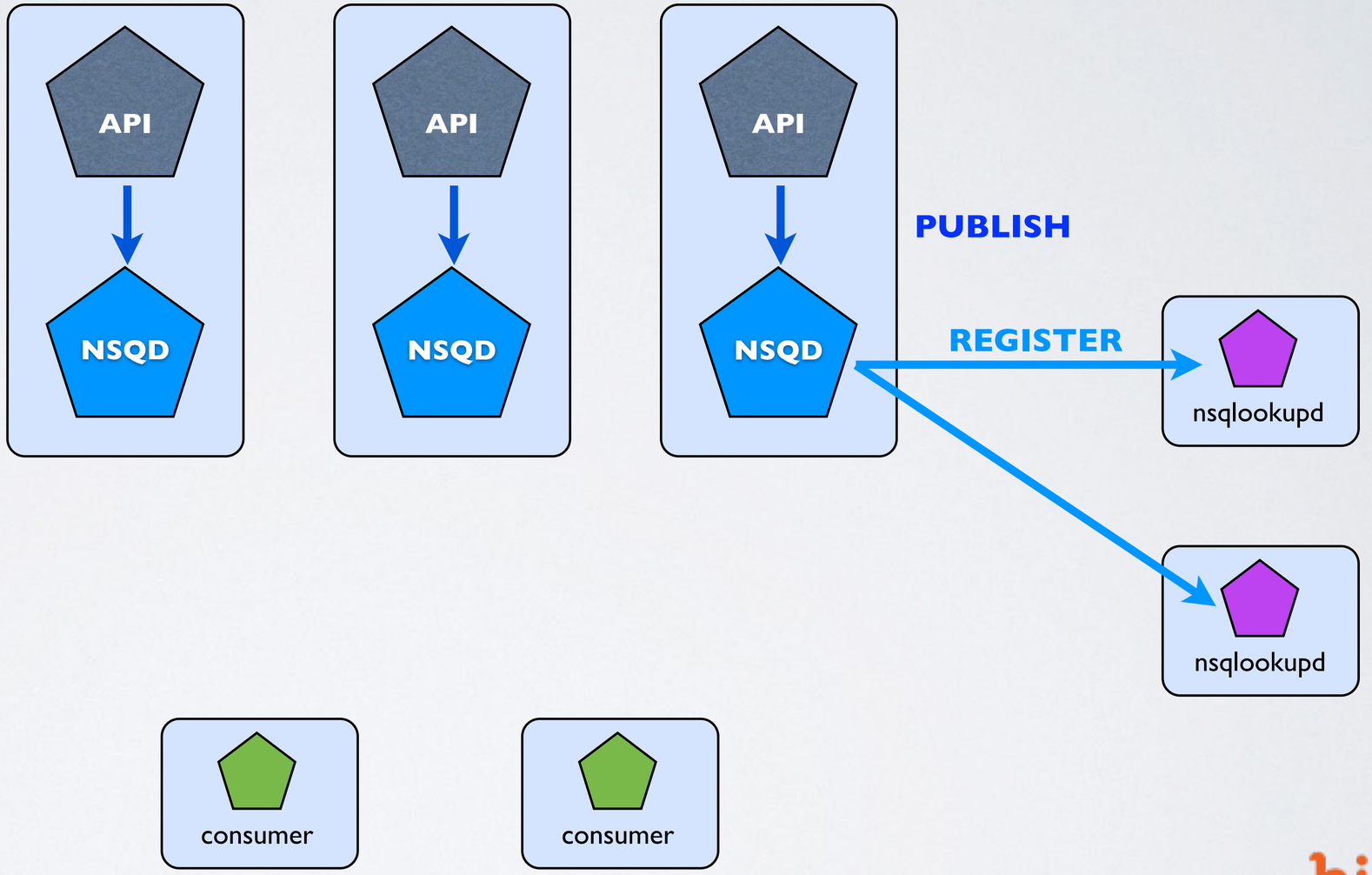
- enable *distributed* and *decentralized* topologies
- no centralized broker
- **nsqlookupd** instances are *independent* (**no coordinatation**)



bitly

TYPICAL NSQ CLUSTER

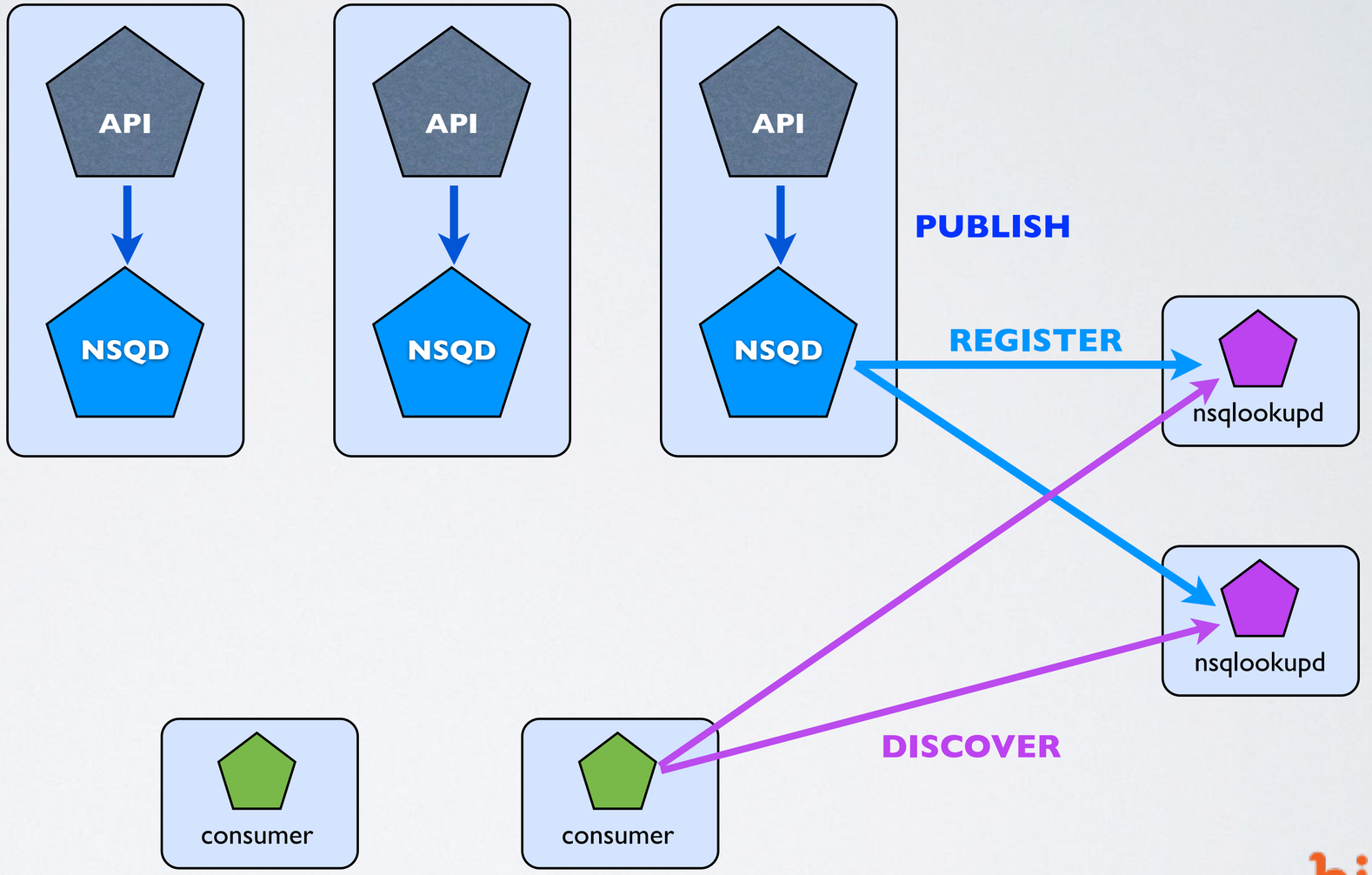
- enable *distributed* and *decentralized* topologies
- no centralized broker
- **nsqlookupd** instances are *independent* (**no coordinatation**)



bitly

TYPICAL NSQ CLUSTER

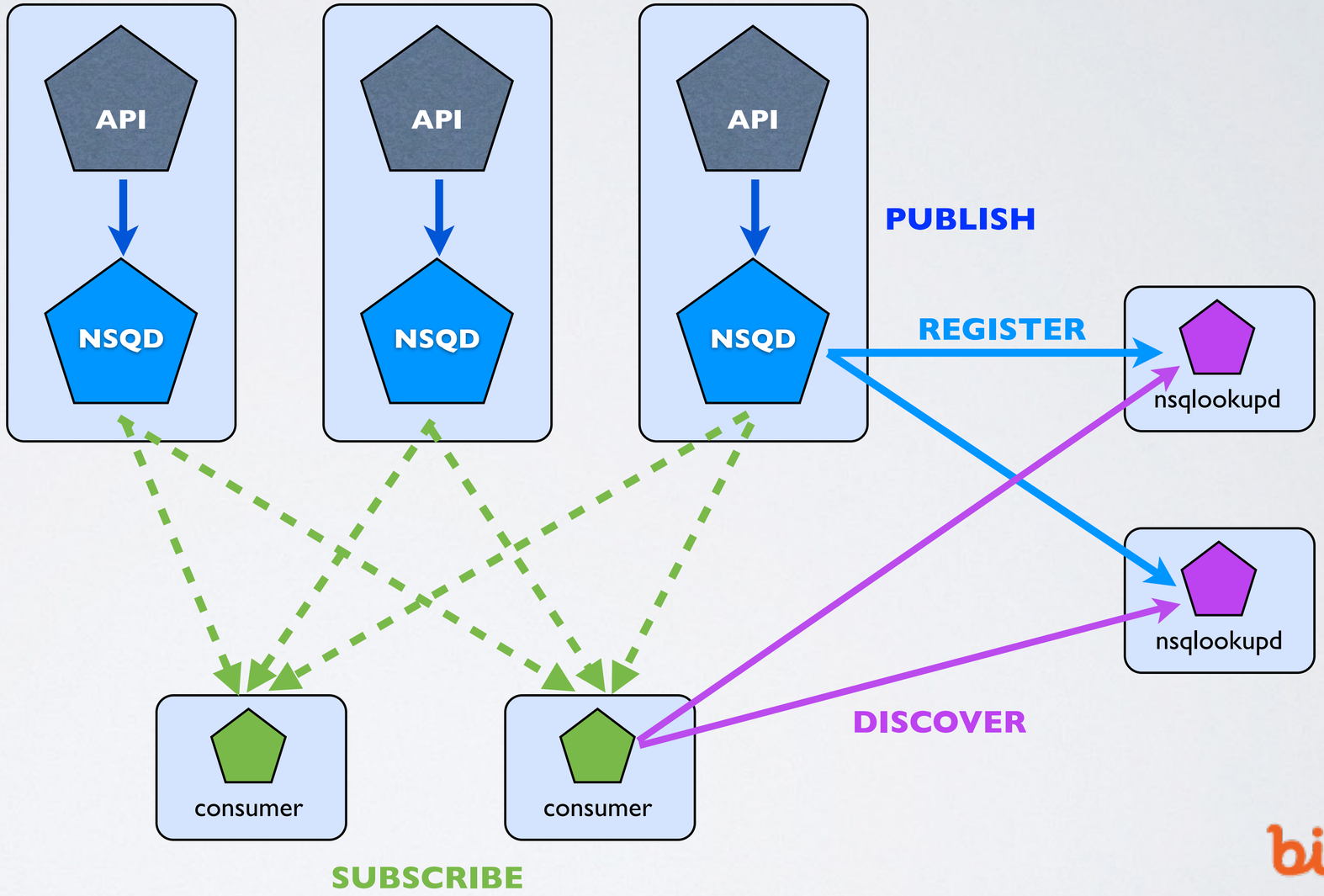
- enable *distributed* and *decentralized* topologies
- no centralized broker
- **nsqlookupd** instances are *independent* (**no coordinatation**)



bitly

TYPICAL NSQ CLUSTER

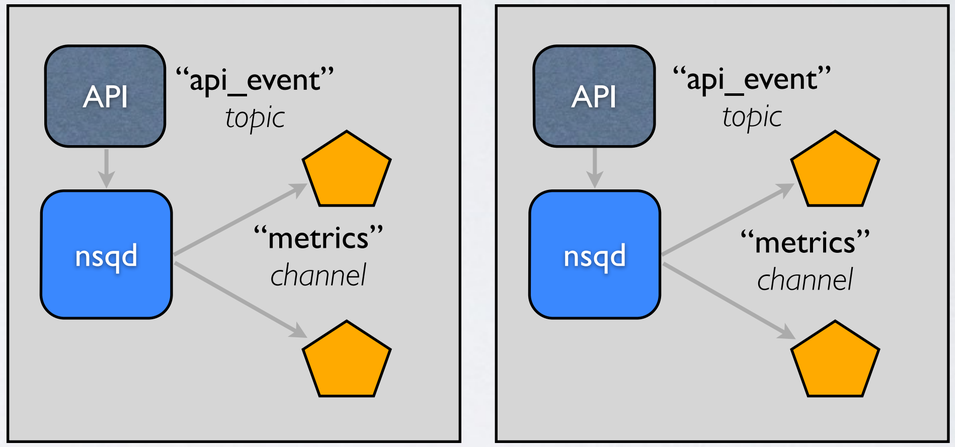
- enable *distributed* and *decentralized* topologies
- no centralized broker
- **nsqlookupd** instances are *independent* (**no coordinatation**)



bitly

SIMPLE EXAMPLE + NSQLOOKUPD

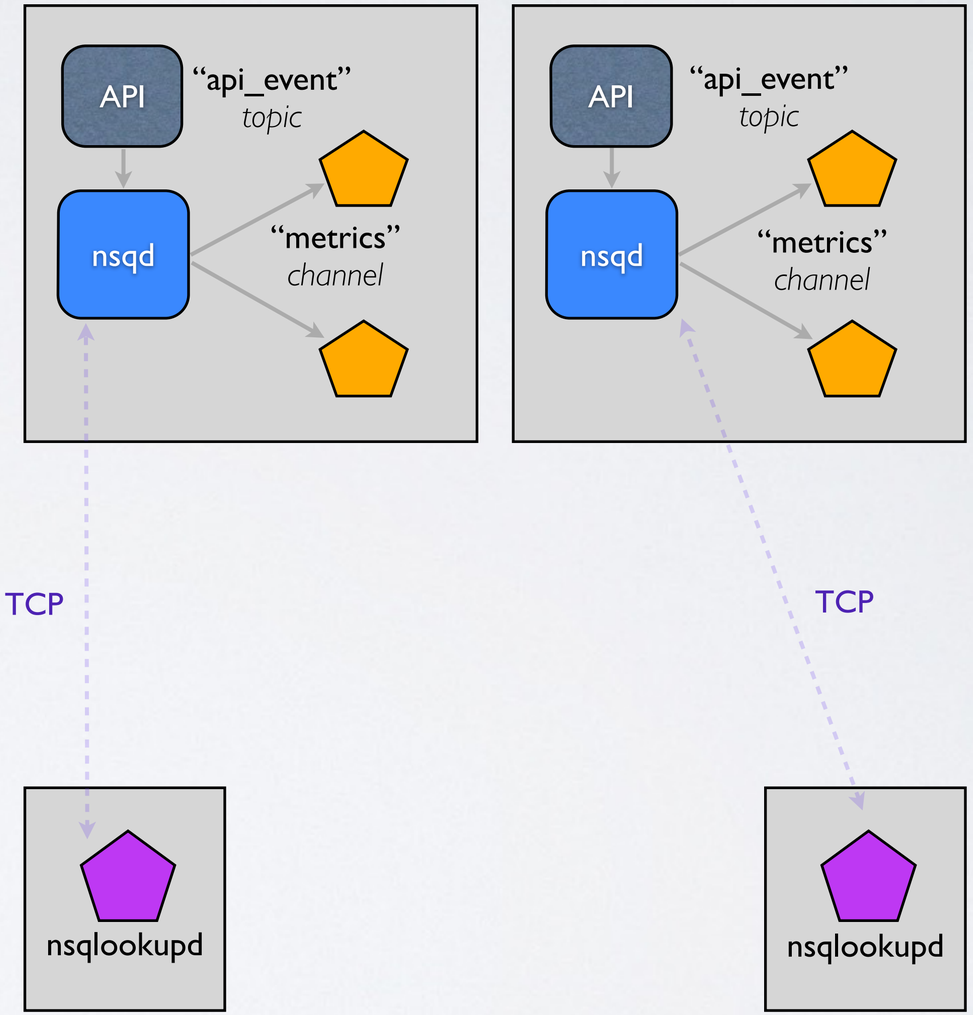
- introduce **nsqlookupd**
- **discoverability**
- producers and consumers *come* and *go*
- other services can *discover* and *subscribe* to this topic



bitly

SIMPLE EXAMPLE + NSQLOOKUPD

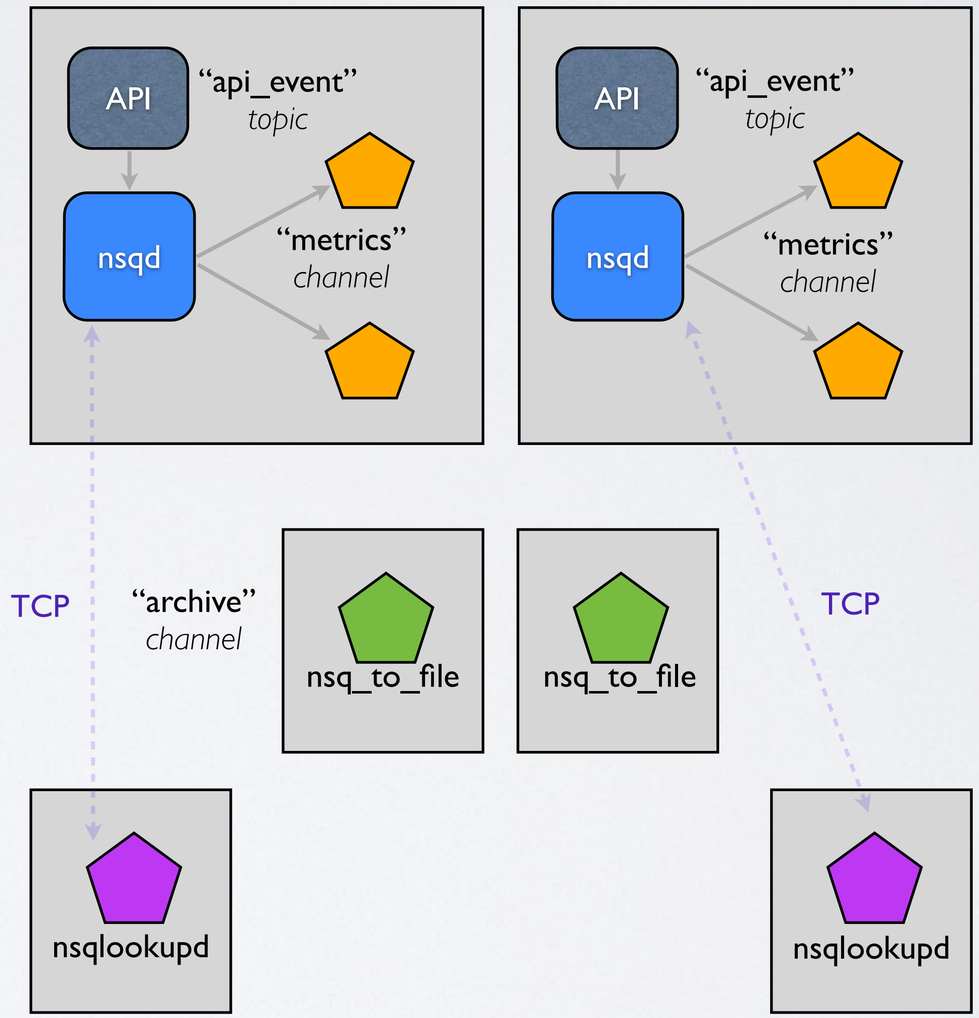
- introduce **nsqlookupd**
- **discoverability**
- producers and consumers *come* and *go*
- other services can *discover* and *subscribe* to this topic



bitly

SIMPLE EXAMPLE + NSQLOOKUPD

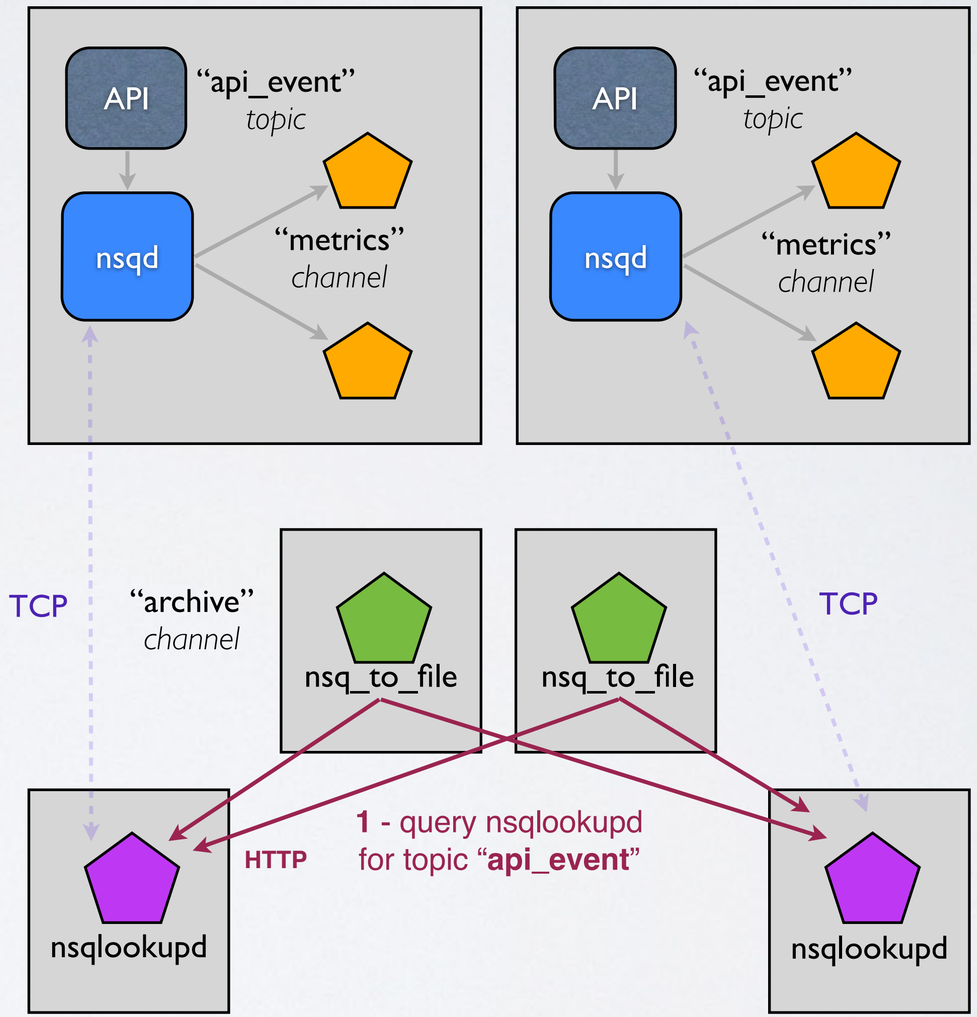
- introduce **nsqlookupd**
- **discoverability**
- producers and consumers *come* and *go*
- other services can *discover* and *subscribe* to this topic



bitly

SIMPLE EXAMPLE + NSQLOOKUPD

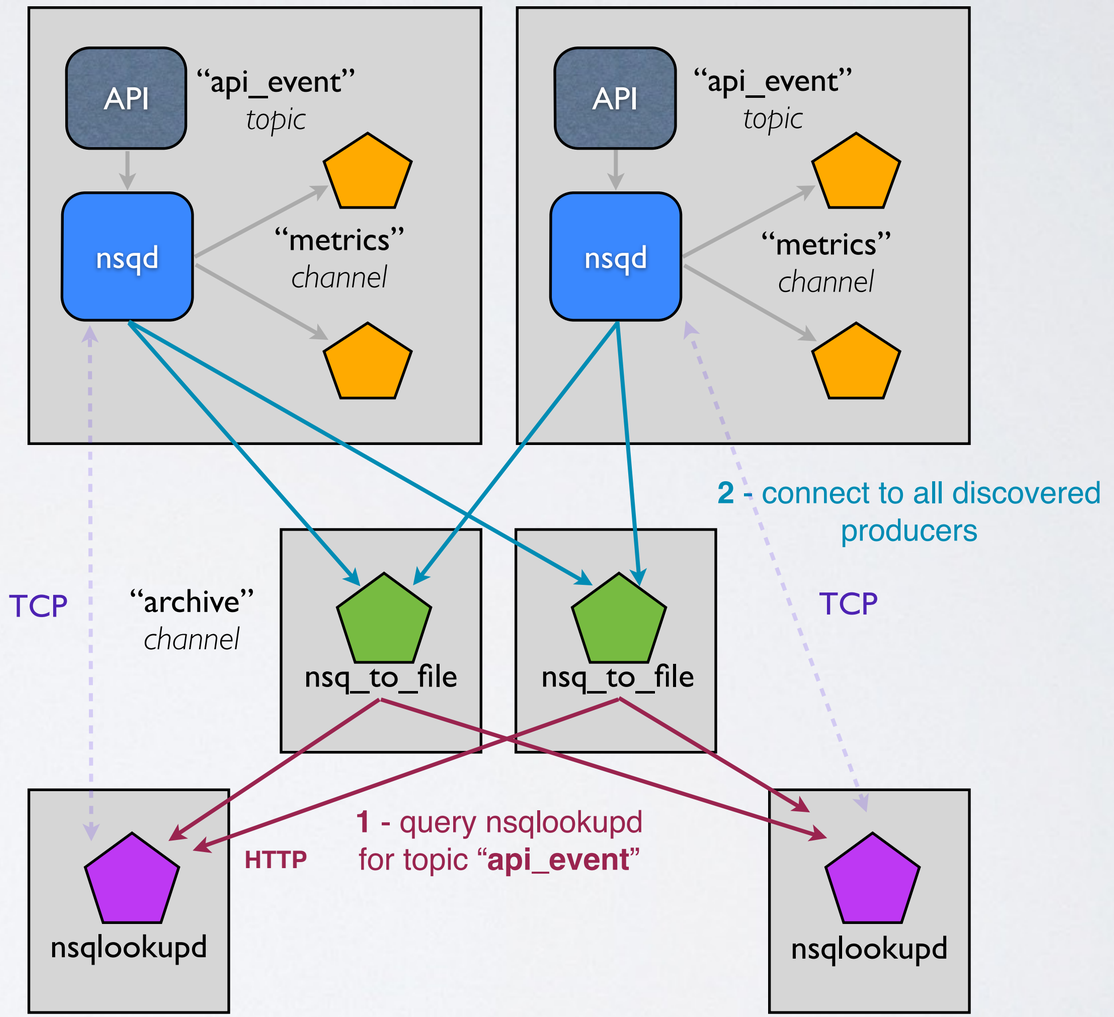
- introduce **nsqlookupd**
- **discoverability**
- producers and consumers *come* and *go*
- other services can *discover* and *subscribe* to this topic



bitly

SIMPLE EXAMPLE + NSQLOOKUPD

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



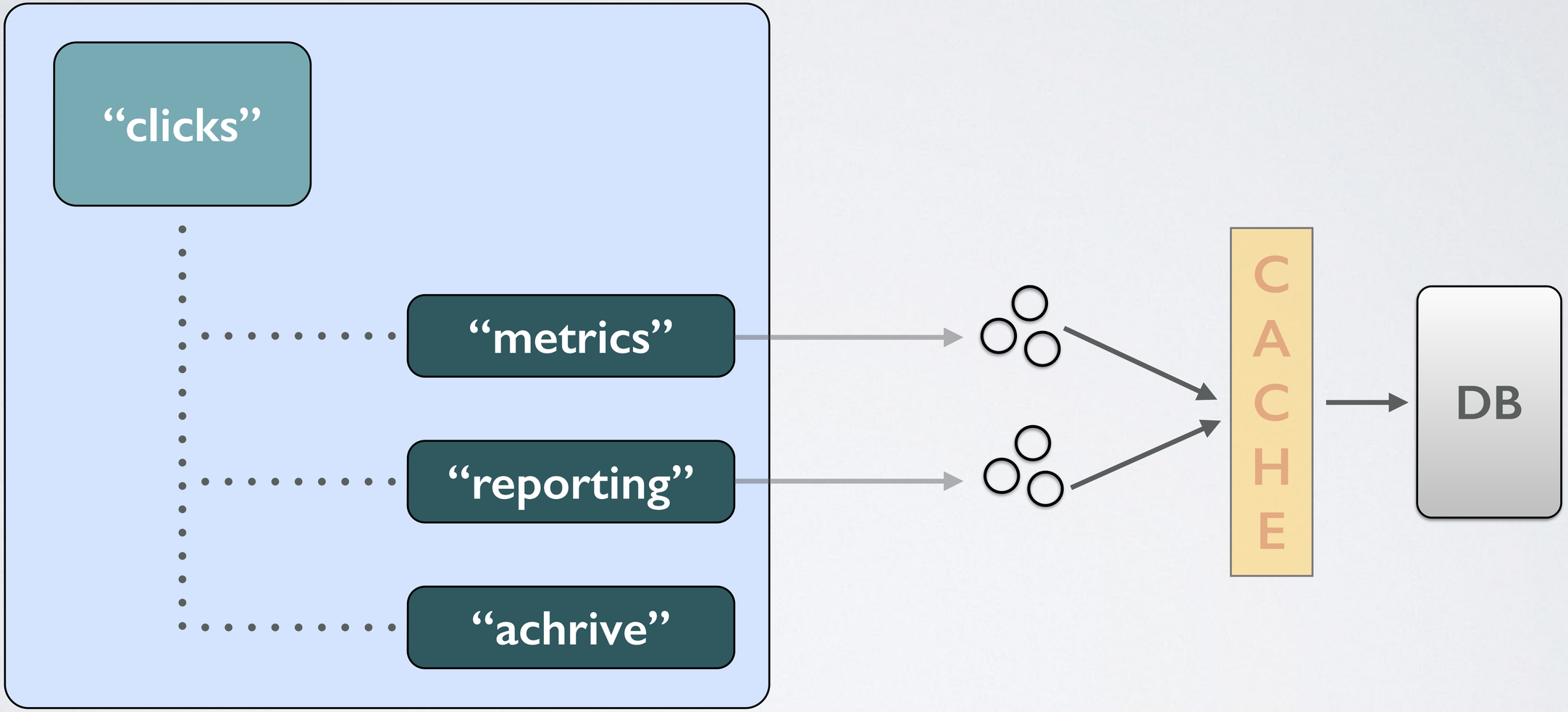
bitly

GUARANTEES

- messages are delivered ***at least*** once
- messages are ***not*** durable (by default)
- messages received are ***un-ordered***
- consumers ***eventually*** find all topic producers

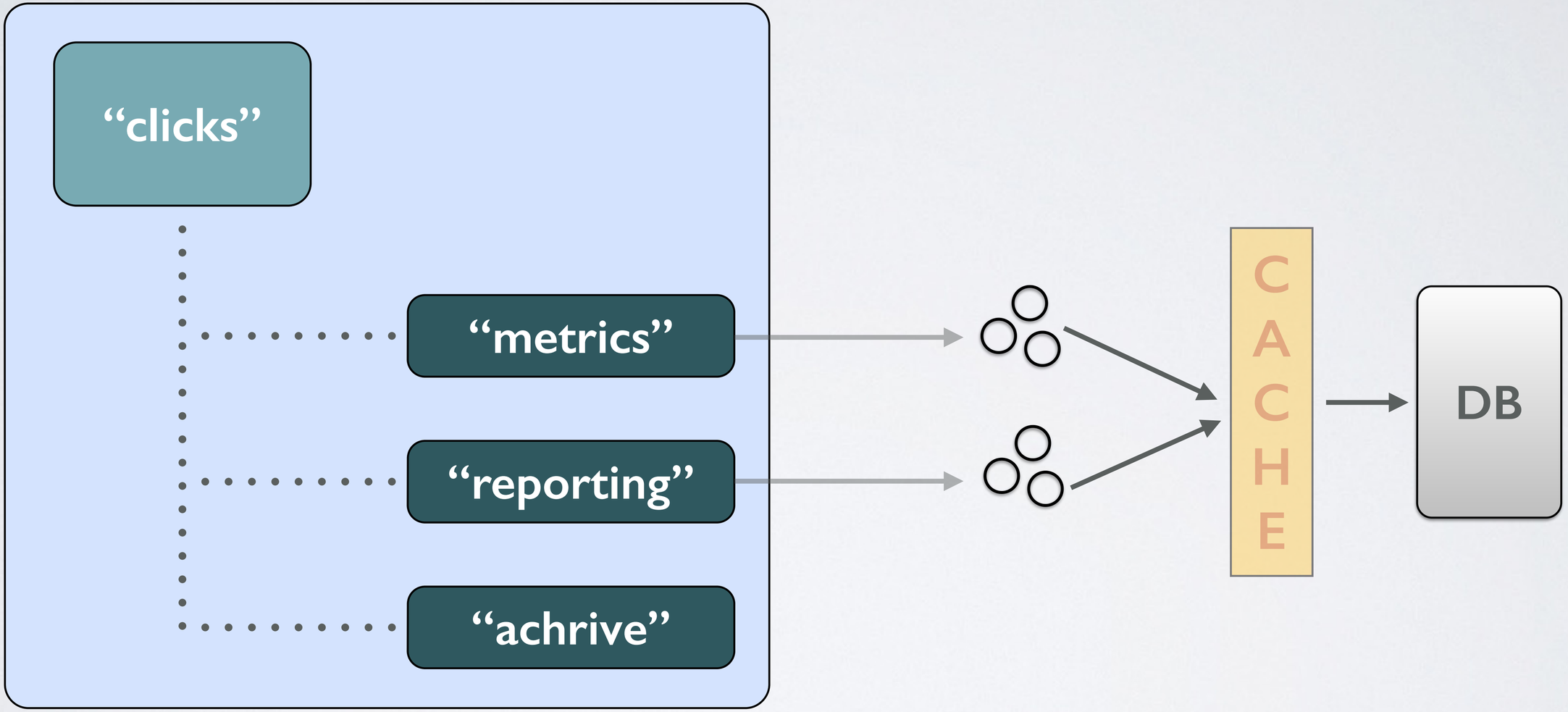
bitly

TOPIC & CHANNEL PAUSING



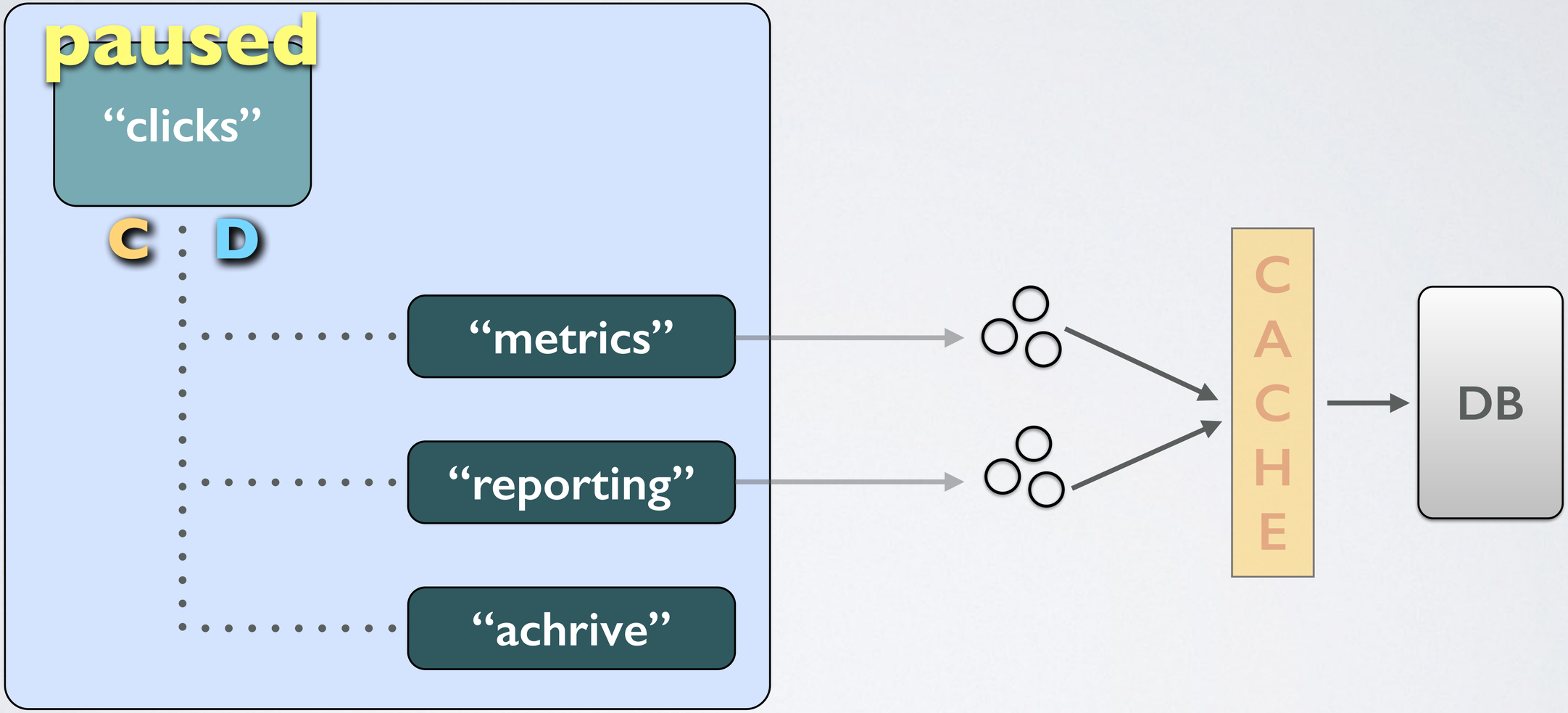
bitly

TOPIC & CHANNEL PAUSING



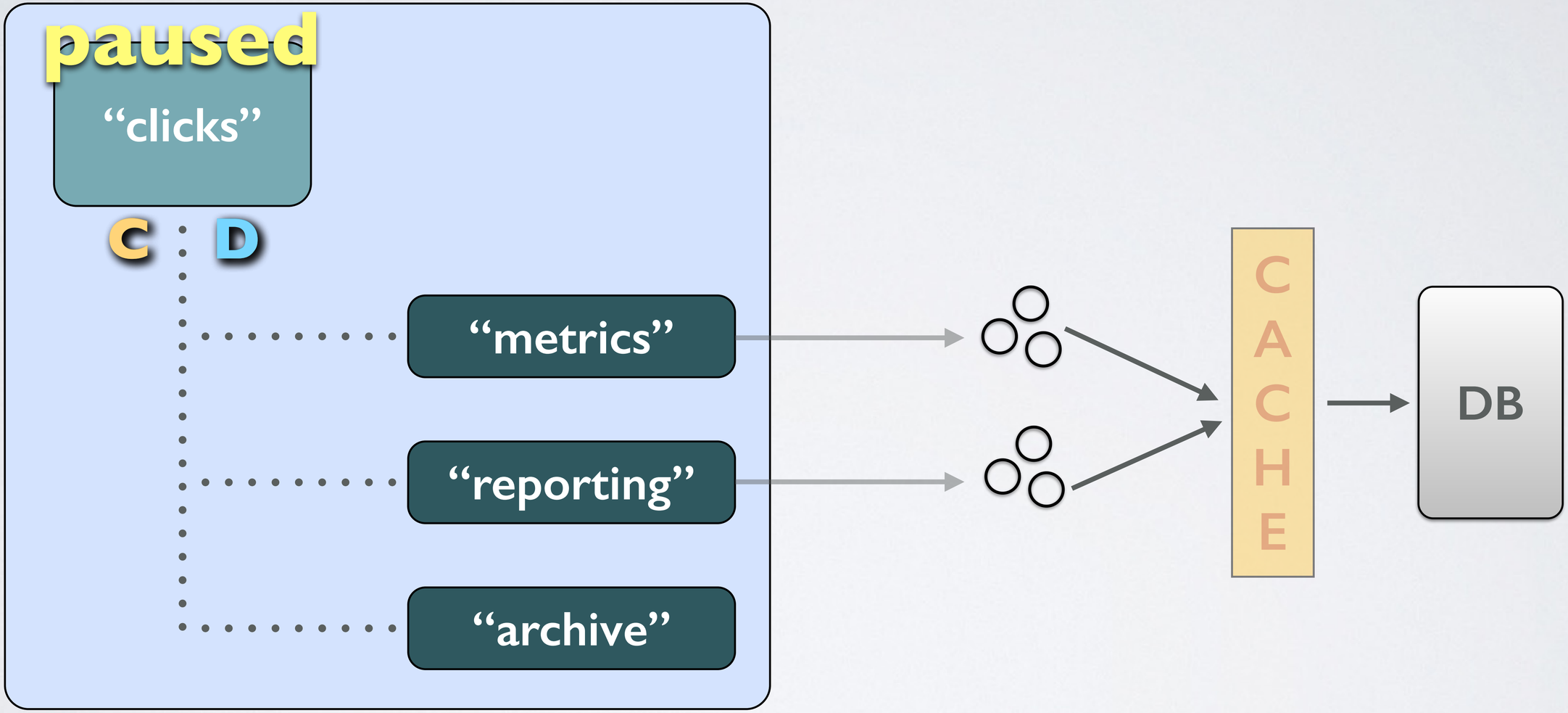
bitly

TOPIC & CHANNEL PAUSING



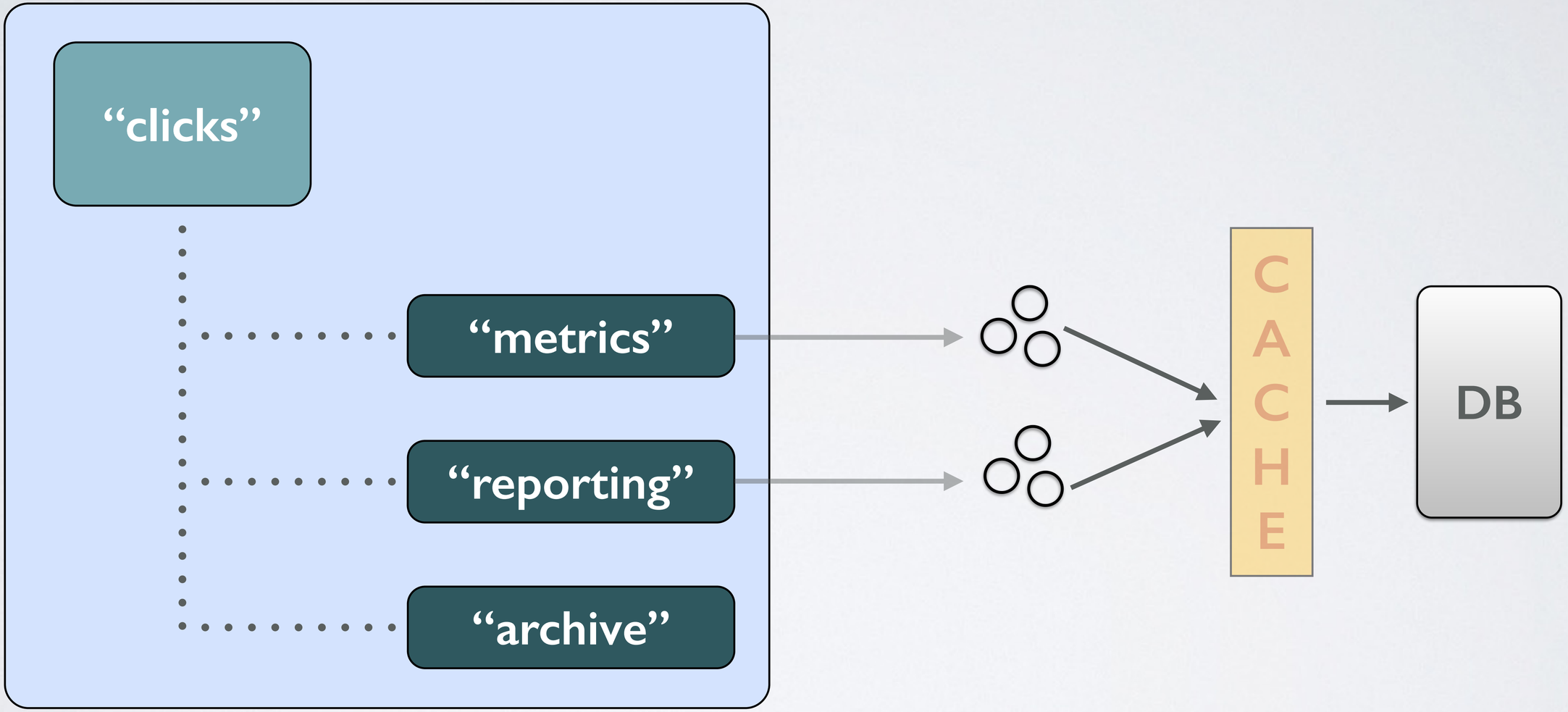
bitly

TOPIC & CHANNEL PAUSING



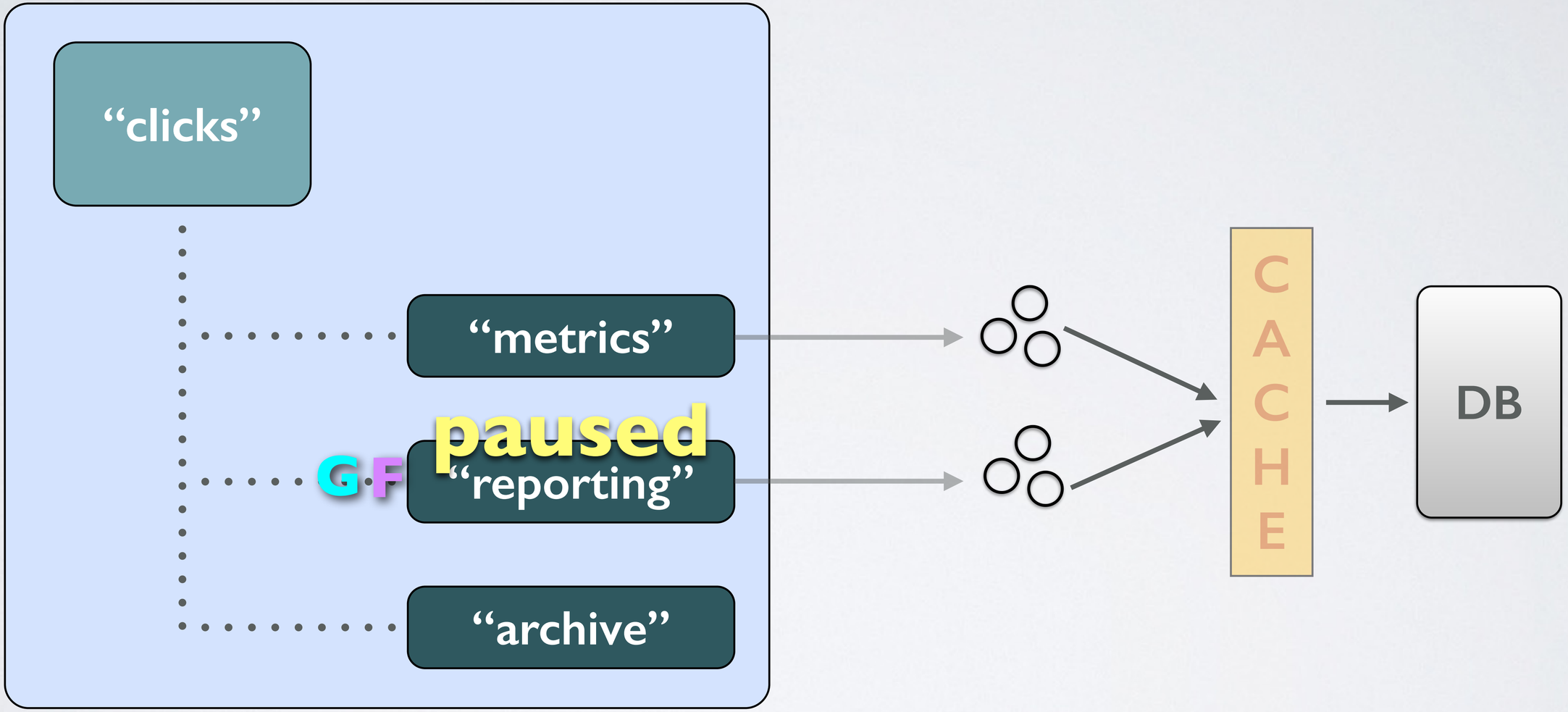
bitly

TOPIC & CHANNEL PAUSING



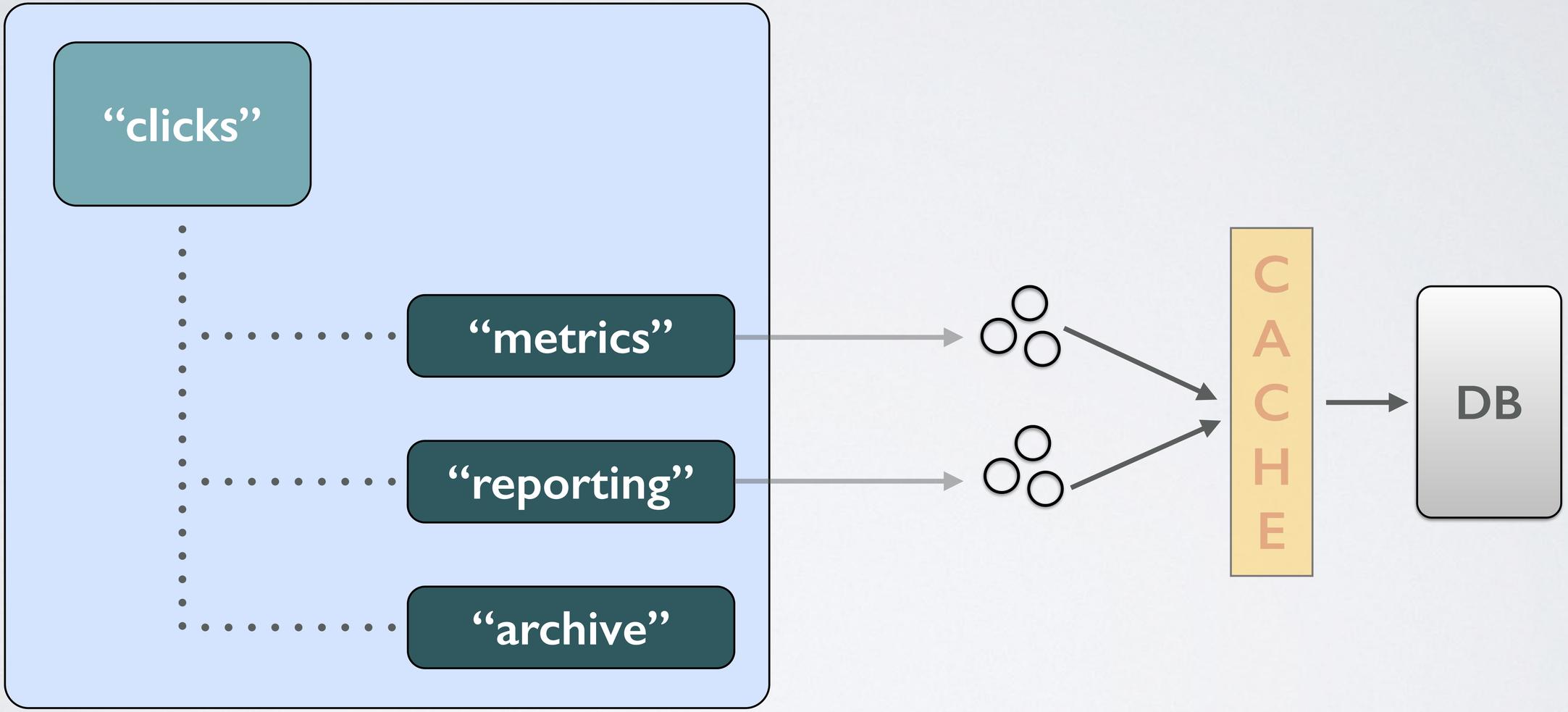
bitly

TOPIC & CHANNEL PAUSING



bitly

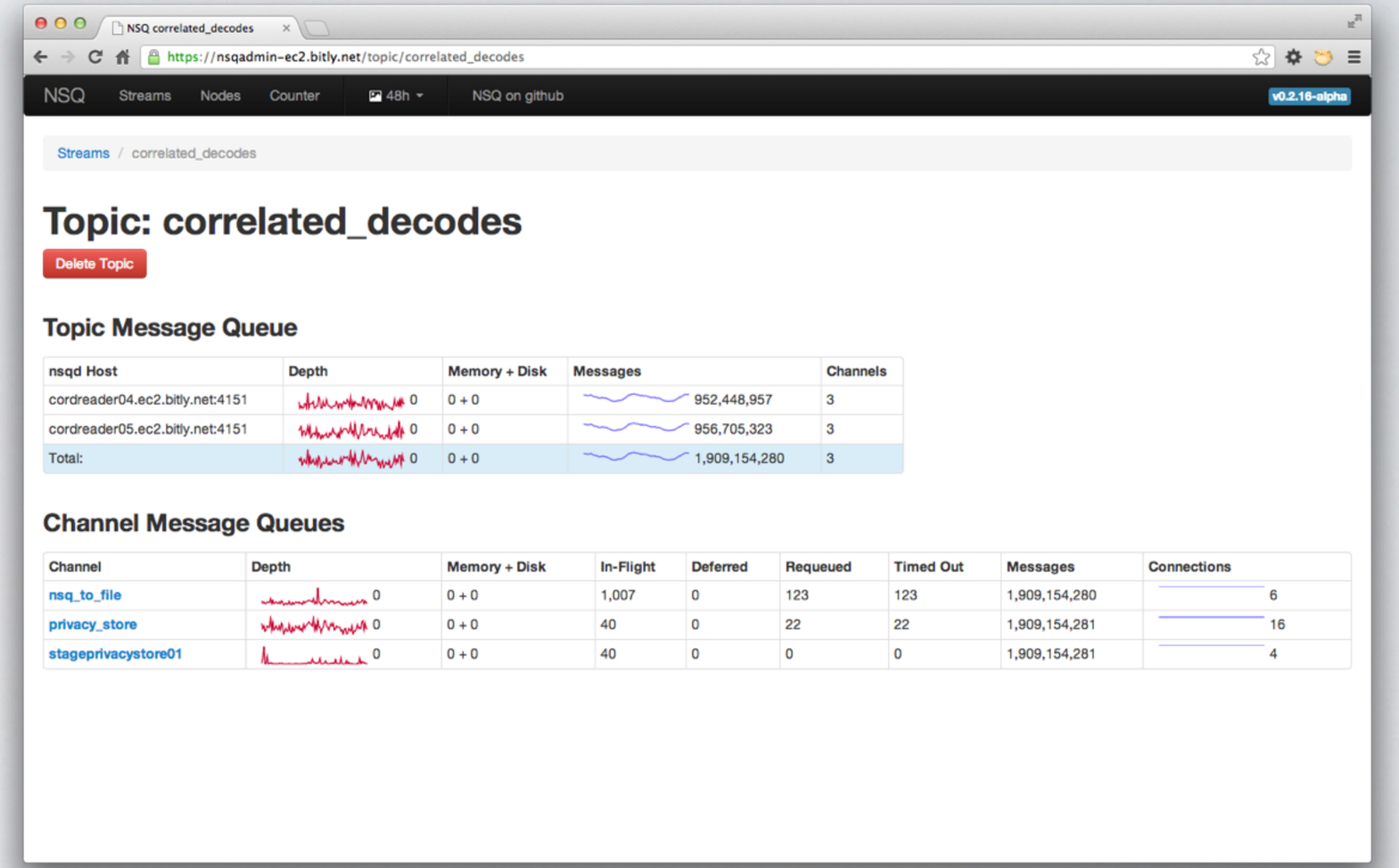
TOPIC & CHANNEL PAUSING



bitly

NSQ ADMIN

- **#ephemeral** channels
  - disappear when last client disconnects
  - useful for scripts/ debug



bitly

Let's install NSQ locally and write some code

http://bit.ly/go-nsq-workshop

bitly

C’est ça!

big thanks to ***@jehiah*** *and* ***@imsnakes***
(authors of NSQ)

bitly