



AN INFLUXDATA CASE STUDY

# How FuseMail Keeps Their Secure Email Service Performant with InfluxDB

**Dylan Ferreira**

Lead Systems Administor, FuseMail



APRIL 2018

# Company in brief

FuseMail®, the email division of j2 Global®, is an advanced managed email solutions provider that, for more than 20 years, has been ensuring the secure and reliable delivery of corporate email messages. Providing email solutions to more than 25 million mailboxes, FuseMail offers a range of cloud-based email security services including sophisticated spam and virus filtering, email archiving, encryption, continuity, hosted Microsoft® Exchange and hosted webmail. FuseMail provides award-winning, comprehensive solutions that are expertly delivered and supported by dedicated teams in Canada, the United States, the UK and Ireland ensuring clients receive optimal performance and guaranteed peace of mind.

# Case overview

FuseMail needed to collect and act on email latency to keep their cloud-based secure email service performant. Their current queue threshold monitoring did not provide visibility into live traffic or system performance specifics. To address this, they built an email latency collection system and used InfluxDB to store the high cardinality metrics gathered from their custom mailers to the tune of over 1.5 million time series per database. Using InfluxDB for event logging storage, and Grafana for data visualization, FuseMail surpassed their monitoring goals by gaining unexpected visibility, control, and debugging capabilities.



Data monitoring and maintenance leveraging raw data

*“We relied on queue depth to detect delivery lag, which shows you when you have really big problems, but it doesn't show you any of the little problems. There were lots of small issues bubbling under the radar.”*

**Dylan Ferreira**, lead systems administrator

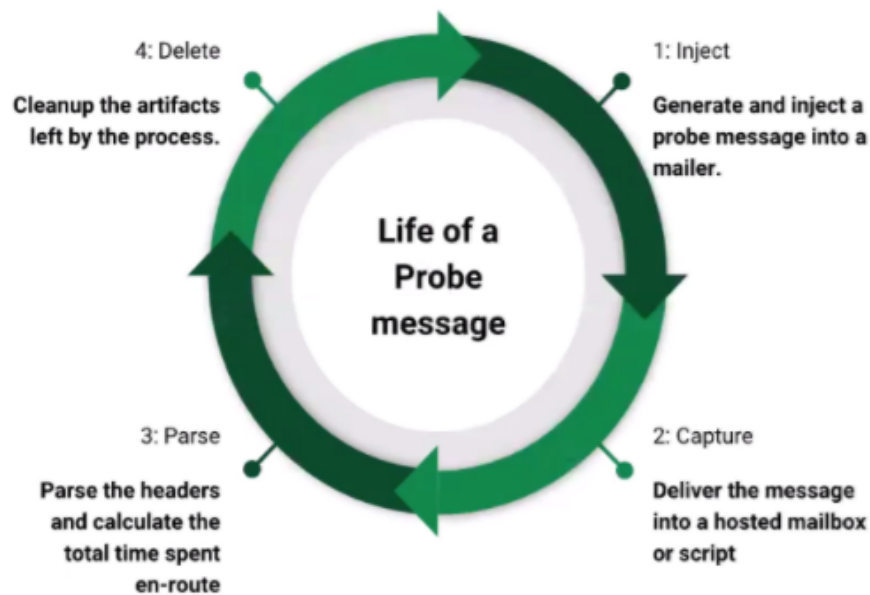
## The business problem

The FuseMail technical team was assigned the task of collecting email latency statistics on their core mailer. The timing data is confined to the time their mailer spends processing messages (scanning, filtering, and routing). The requirements were that they keep long-term latency percentile data in daily, weekly, and monthly aggregates so that they can report their performance and track it against their SLA, review their software updates and deployments, check for changes in overall performance, and improve capacity planning. The faster they could identify spam and a compromised account or system, the better their email security solutions and customer experience.

## The technical problem

FuseMail had relied on queue threshold monitoring to detect delivery lag. This monitoring method indicates overall system health but does not provide in-depth latency insights or insights about actual latency in production. Further, relying on a single metric like queue depth makes it impossible to tell the difference between an increase in message rate or a timeout caused by an unhealthy dependency. The initial idea was to send probe messages through their mailer: periodically send messages through the platform, catch them on the other side, and track how long it took for them to go through.

## The Probe Message



The probe message approach presented the problems of maintaining lists of hosts, maintaining special probe accounts, the difficulty of generating probes to test all possible routes through the platform, and the fact that probes add extra load. Rather than testing in an isolated and artificial environment, FuseMail decided to measure actual production traffic. Around the same time that they were assigned this project, their mailer team was working on building event logging directly into their mailer.




The mailer would publish information on deliveries into NSQ (a simple distributed message bus) for consumption by various downstream services. Upon realizing they could tap into this to gather stats on their production traffic, their mailer team agreed to add high-precision timing data to the payload, enabling them to get message counts and, effectively, a new form of logging that allowed them to zero in on customer experiences. To store that data, they set out to search for a time series database.

## The solution

*“With InfluxDB, we had all the benefits we saw in Prometheus with labels, without the tradeoffs of immediately aggregating. And on top of this, we could store multiple fields per row, so lots of useful context could be stored along with the metrics.”*

## Why InfluxDB?

FuseMail chose InfluxDB because it can store raw metrics, it does not aggregate data on ingest, and it can store multiple fields per row. Each of InfluxDB, Prometheus and Graphite have a place in FuseMail architecture, but for any event logging, FuseMail uses InfluxDB.

	<ul style="list-style-type: none"><li>• Can store raw metrics.</li><li>• Flat structure with metadata stored in the metric name (pre-v1).</li><li>• Fixed per-series aggregations</li></ul>
	<ul style="list-style-type: none"><li>• Multi-dimensional TSDB.</li><li>• Very high performance.</li><li>• All data is aggregated on ingest.</li></ul>
	<ul style="list-style-type: none"><li>• Can store raw metrics.</li><li>• Multi-dimensional TSDB.</li><li>• Complex aggregations (CQs).</li><li>• Multiple fields!</li></ul>

Upon choosing InfluxDB, FuseMail performed load testing on it to determine the:

- Speed at which to push data into InfluxDB with the hardware they had available
- Size of storage they would need
- Schema design in cardinality (selecting tags and fields)
- Optimum InfluxDB config and ingest config for the data
- Impact of Continuous Queries (CQs) on the database while under load

Among the InfluxDB load testing lessons learned were:

- Batched posts to the HTTP-API are fast and reliable whereas UDP, used alongside an ephemeral NSQ channel, caused data loss anytime the host was under pressure.
- Continuous Queries (queries that run automatically and periodically on real-time data and store query results in the specified measurement) needed to be modified with an overlapping look-back to fill in whatever might not have made it in by the time they did their Continuous Query.
- Early dashboard mockups were valuable to understand exactly what data to collect and what aggregations are needed.

## Early Dashboard Mockup with Random Test Data



In working with InfluxDB, FuseMail relied on `nsq_tail` to consume data since their mailers produce event log data into NSQ. To transform or pre-filter the data in any way, they piped it through `jq`, a lightweight and flexible command-line json processor. To work with json formatted data, they built what they call `json_to_influx`. It takes a stream of json in payloads on STDIN, outputs InfluxDB line protocol to the HTTP API, and uses a config to match it to a schema. FuseMail refer to this piece of their architecture (`nsq_tail`, `jq`, `json_to_influx`) as the “consumer”.

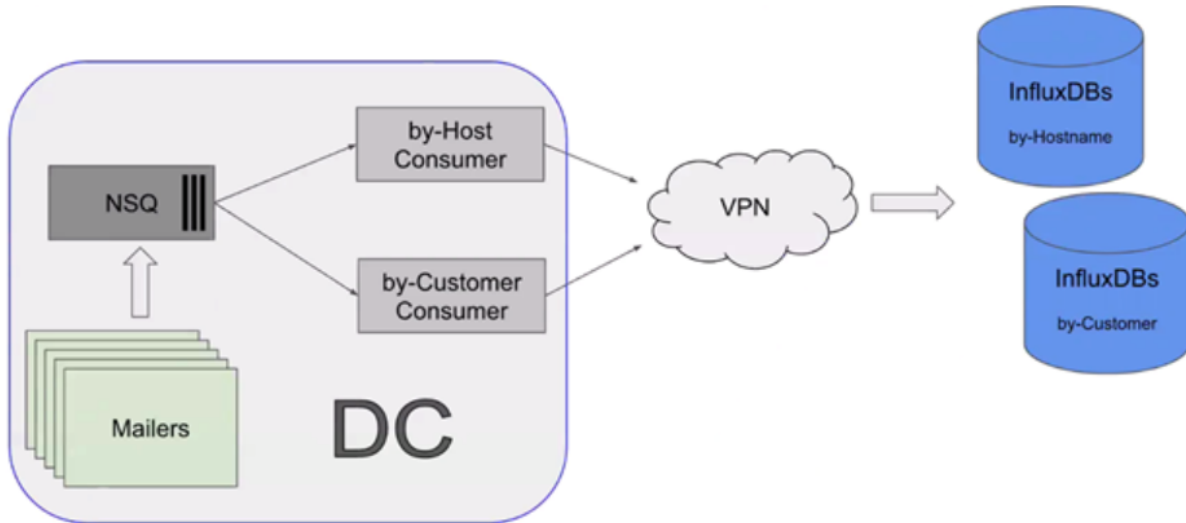
## Why Kapacitor?

FuseMail had explored Kapacitor in R&D work, pushing event log data into it out of their mailers and writing TICKscripts to filter and aggregate the data down and to spot compromised accounts or spam. The TICKscripts would aggregate the data into views such as recipient address count by sender address, hello count by sender address, or sender domain count by IP.

FuseMail decided to use Kapacitor, InfluxData’s real-time streaming data processing engine, in a redesign of their latency metrics setup (v.2.0) to meet the requirement of General Data Protection Regulation (GDPR) that was coming into effect. GDPR prohibits export of Personally Identifiable Information (PII) such as email addresses, IP addresses, etc., out of a user’s region. Yet FuseMail’s model required that they maintain historical data centrally, so they redesigned their approach to collect and act on the data locally and adhere to the GDPR data requirements locally, then send the aggregation back to the central databases to maintain a historical view. They did this by writing the raw event log data in each data center to a local InfluxDB in that datacenter that had local retention policies and then using Kapacitor to perform batch queries, so they can aggregate the local data centrally.

# Technical architecture

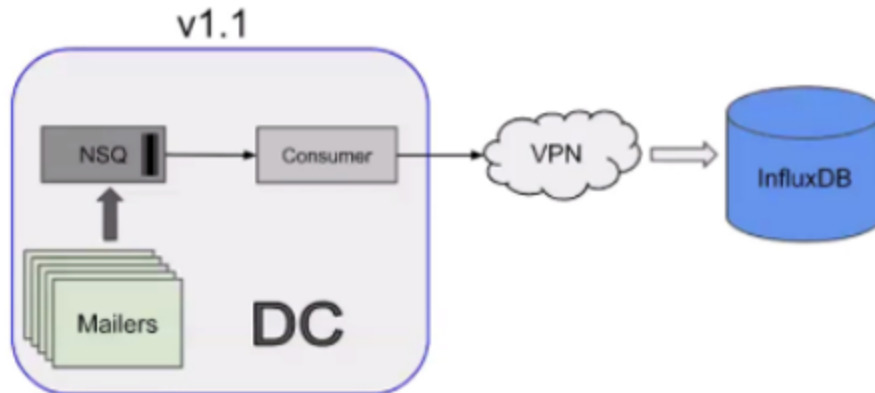
InfluxDB in v1.0 of FuseMail's Email Latency Metrics Setup



In FuseMail's v.1 of the email latency collection system, the dataflow was as follows:

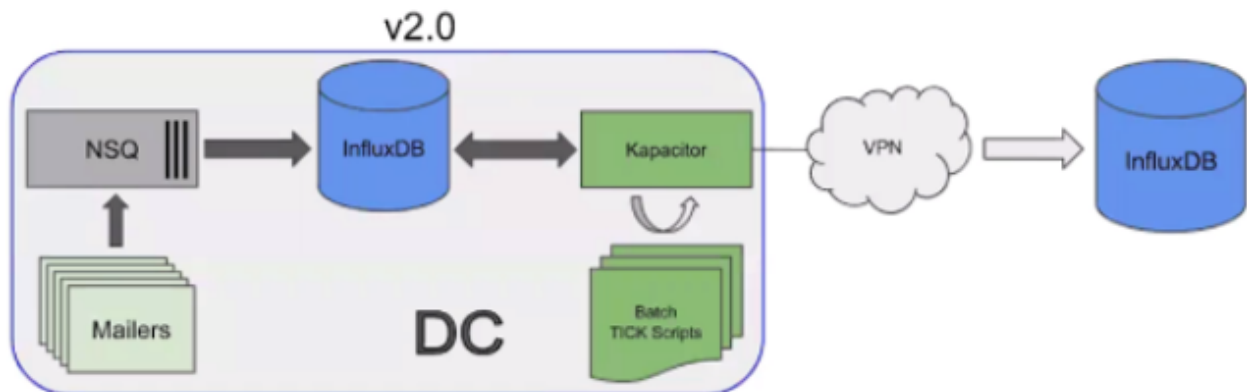
- The mailers pushed data into NSQ.
- A set of "consumers" in each DC (a total of nine data centers) batch-posted into a central InfluxDB database instance, where Continuous Queries aggregate the data down.
- FuseMail ran dashboards directly from this instance.
- At the time they created v.1, the database couldn't handle the combined cardinality of hosts and customers. So they ran two of these rigs (databases): one tagged by data center, hostname, and several other low-cardinality, low-priority tags, and one tagged by data center and customer, which provided a view into customer-specific experience problems.

## InfluxDB in v1.1 of FuseMail's Email Latency Metrics Setup



When v1.1 of FuseMail's email latency collection system came out, InfluxData had released the Time Series Index (TSI)—which supports a large number of time series, i.e., a very high cardinality in the number of unique time series that the database stores. So FuseMail found that a single database could easily handle all of the cardinality (a cardinality of 1.75 million at the time).

## InfluxDB and Kapacitor in v2.0 of FuseMail's Email Latency Metrics Setup



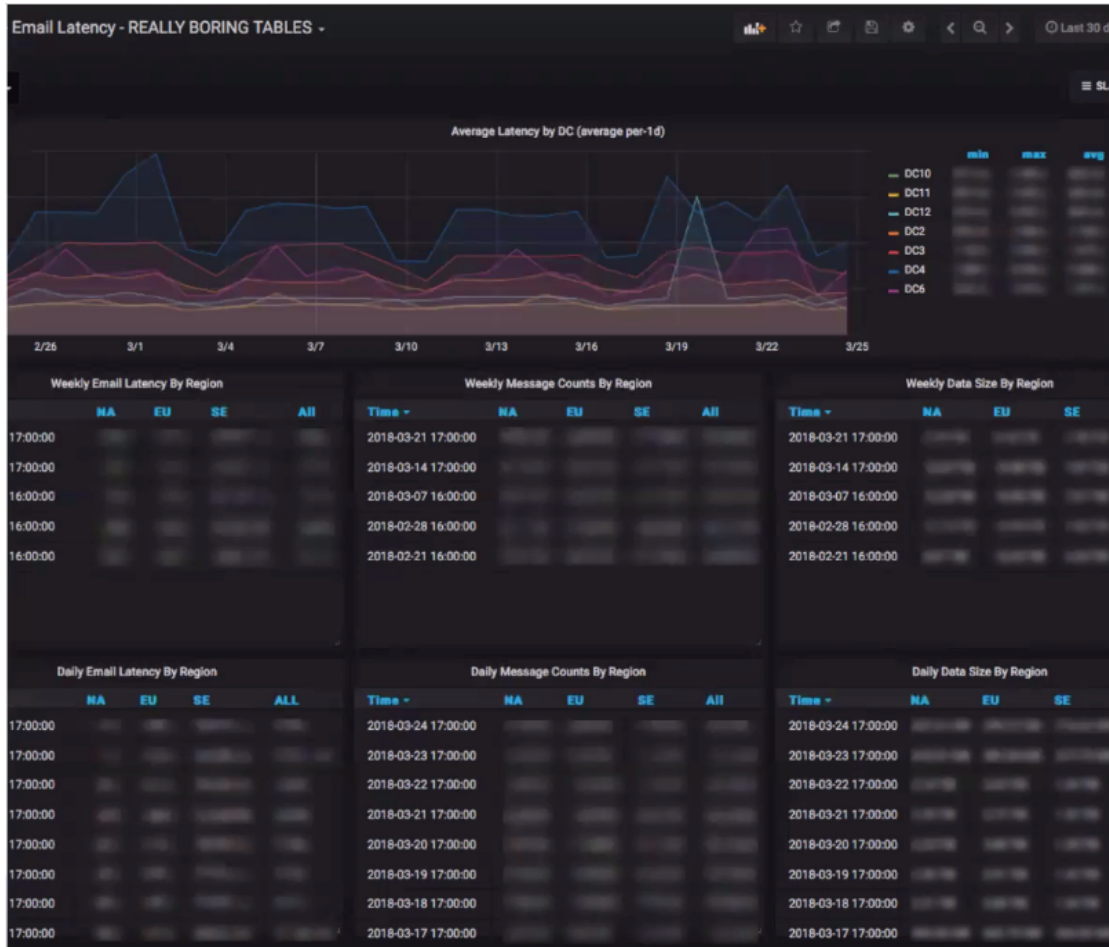
In v2 of FuseMail's email latency metrics setup integrating Kapacitor, the dataflow is as follows:

- The per-DC InfluxDB databases hold IP addresses, but with tighter retention policies, while the central InfluxDB database gets high-level aggregations and drops all PII for long-term storage.
- This model scales better as the per-DC databases only have to cope with the raw data from a single DC, while the central database only has to cope with ingesting the long-term aggregations.
- Sensitive data doesn't have to leave the data center, and FuseMail still get their long-term percentiles and counts in a central place that's easy to query.
- InfluxDB retention policies can be set locally to help fulfill GDPR requirements.



To provide the latency stats for which the project was initially launched, FuseMail would get a daily or weekly rate, have InfluxDB aggregate hourly, have Grafana do a sum query of InfluxDB of hourly aggregates, and then lock aggregations to specific date boundaries (such as week or month).

### Aggregations for Reporting Purposes



## What's next for FuseMail?

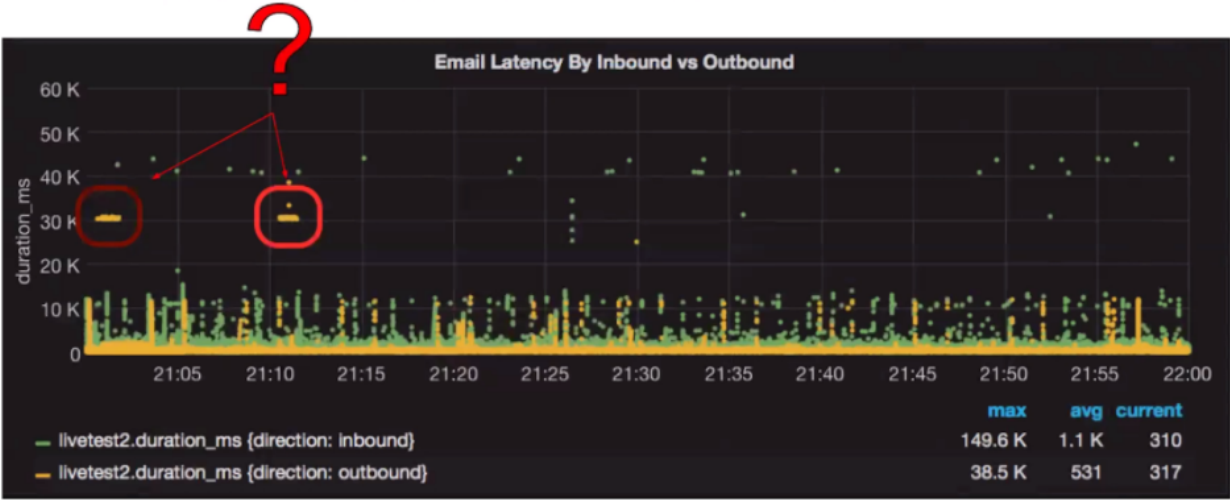
FuseMail plans to further explore Kapacitor's anomaly detection capabilities (to detect compromised accounts and incoming spam waves) as well as Telegraf's metrics collection capabilities. Telegraf, which now has an NSQ "consumer" that can take json in and transform it into line protocol into Influx, can now replace FuseMail's json\_to\_influx.

# Results

*“When I began, I thought I'd be spending most of my time figuring out how to get data into InfluxDB efficiently, but once that was taken care of, I found myself spending even more time looking at the data and finding new ways to express it.”*

Looking at the raw data over the first few weeks, FuseMail started seeing interesting patterns: micro outages and timeout retry patterns which revealed underlying config problems, query issues, and hardware problems that had gone unnoticed by threshold monitoring. They started to recognize distinct patterns in the data, like glyphs or signatures of specific types of problems. They realized that by examining raw data scatterplots, they could tune everything in their stack.

Unexpected Advantages of Raw Data



FuseMail built a dashboard for looking at email delivery issues that were difficult to detect from the logs and called it “Warhol Vision” (recalling Andy Warhol paintings).

## Warhol Vision



The dashboard features four large panels running nearly the same query, but each panel shows the data tagged by one of four main tags:

1. Direction – inbound versus outbound traffic
2. Data center – which data center that traffic was in
3. Mailer – which mailer or which host that traffic was in
4. Customer – which customer the messages belong to

The dashboard shows all critical dimensions of the data at once and thereby the scope of an issue to debug it faster.

Because FuseMail measured live traffic, they started to use all the raw data to debug their environment. Having all the raw data they could get with InfluxDB, as opposed to an aggregated system, proved extremely valuable because it provided unexpected visibility, to the extent that the actual SLA monitoring requirements became a side issue.

## Other Interesting Views



FuseMail can now build retrospective dashboards based on the available data, to examine a particular problem. For example, they run a Continuous Query to grab the data for worst customer experiences in the last 24 hours, resulting in a lightweight and useful table. Having the customer numbers as a label means they can find which customers are having problems and address that, rather than probe messages that only indicate system health as a whole.

## About InfluxData

InfluxData is the creator of InfluxDB, the leading time series platform. We empower developers and organizations, such as Cisco, IBM, Lego, Siemens, and Tesla, to build transformative IoT, analytics and monitoring applications. Our technology is purpose-built to handle the massive volumes of time-stamped data produced by sensors, applications and computer infrastructure. Easy to start and scale, InfluxDB gives developers time to focus on the features and functionalities that give their apps a competitive edge. InfluxData is headquartered in San Francisco, with a workforce distributed throughout the U.S. and across Europe. For more information, visit [influxdata.com](http://influxdata.com) and follow us [@InfluxDB](https://twitter.com/InfluxDB).



## Try InfluxDB

Get InfluxDB

Contact us for a personalized demo [influxdata.com/get-influxdb/](https://influxdata.com/get-influxdb/)

