Migrating to OpenTelemetry from a Custom Distributed Tracing Pipeline

Francis Bogsanyi

shopify

Historical architecture

What do you need for tracing?

- Instrumentation (in Ruby for Shopify)
- Context propagation
- Collection
- Backend / UI — storage, search, rendering

What instrumentation existed in 2016?

- ~~Jaeger for Ruby, OpenTracing for Ruby~~
- ~~Zipkin for Ruby~~ (abandoned)
- ~~OpenZipkin for Ruby~~ (nascent)

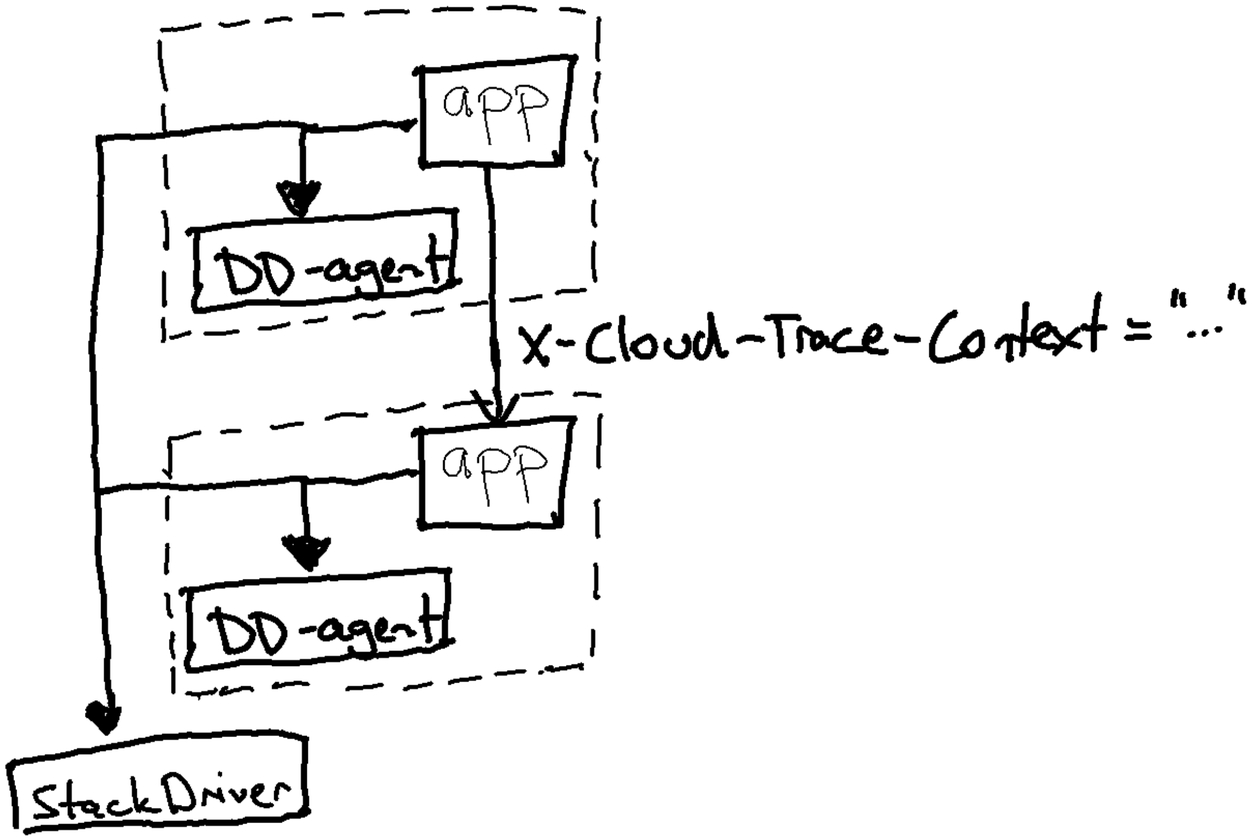
➡️ No obvious trace instrumentation candidate for Ruby

What context propagation existed in 2016?

- ~~W3C traceparent~~
- Zipkin B3 multi-header
- X-Cloud-Trace-Context

What backends/UIs existed in 2016?

- ~~Jaeger~~
- Zipkin
- Datadog
- Google Stackdriver Trace — free, with API rate limits



- Instrumentation loosely inspired by OpenTracing
- X-Cloud-Trace-Context for propagation
- Direct export to Datadog agent and StackDriver

Welcome to Distributed Tracing!

What is this?

This is a PR that will augment your application with distributed tracing. In other words, it will allow you to look at individual requests and follow your application's interactions with external services, such as database, cache, HTTP calls, and will also follow background jobs and annex those interactions to the request that originated them.

Wait, what?!?

Distributed Tracing aims to make your application's interactions with outside services more transparent. This means that every interaction your application has with an external service (be it a database, cache, HTTP API, or potentially others) will be recorded. Inside a request's lifecycle (and any jobs it originates), you will be able to find:

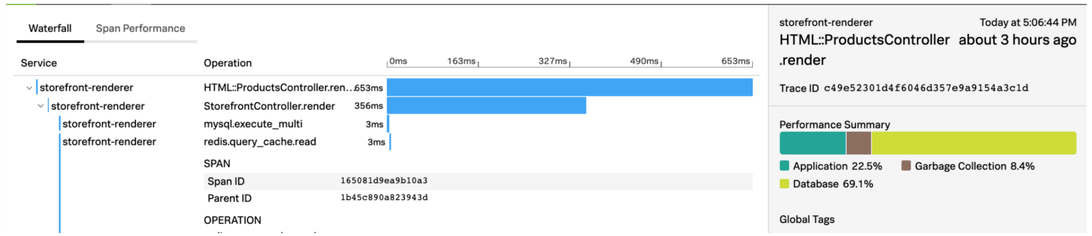
- When the operation started (relative to when the request hit your application) and how long it took
- Details about the operation:
  - For databases, you'll see a query
  - For Redis/Memcached calls you'll see the actual commands
  - For HTTP, you'll see the method, hostname and response code

Out-of-the-box, this gem supports:

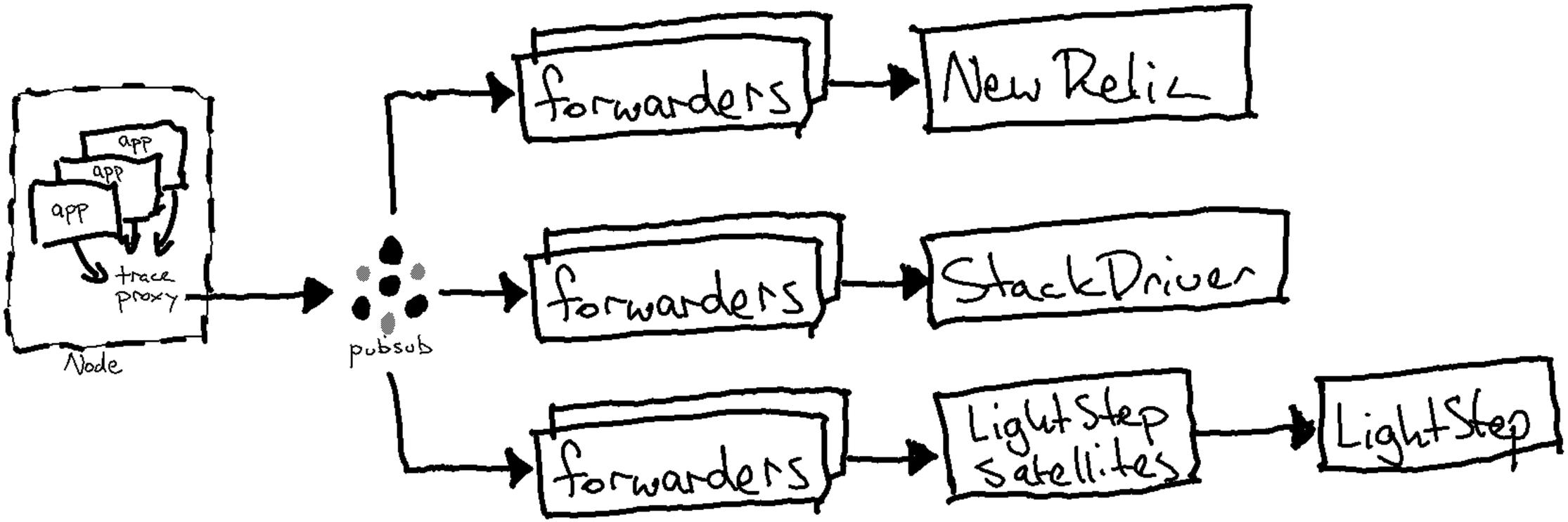
- Ruby on Rails (instruments SQL, Cache and View rendering)
- Memcached (memcached and dalli gems)
- Sidekiq
- Resque
- Net::HTTP
- ElasticSearch (if you use the shopify-elasticsearch gem)
- Rack middleware that you can plug in on non-Rails applications.

For even more details on what Distributed Tracing is, check out our project page.

What does it look like?




app
app
app
trace
proxy
Node
pubsub
forwarders
StackDriver


app
app
app
trace
proxy
Node
pubsub
forwarders
forwarders
forwarders
New Relic
StackDriver
LightStep
satellites
LightStep

What’s wrong with this architecture?

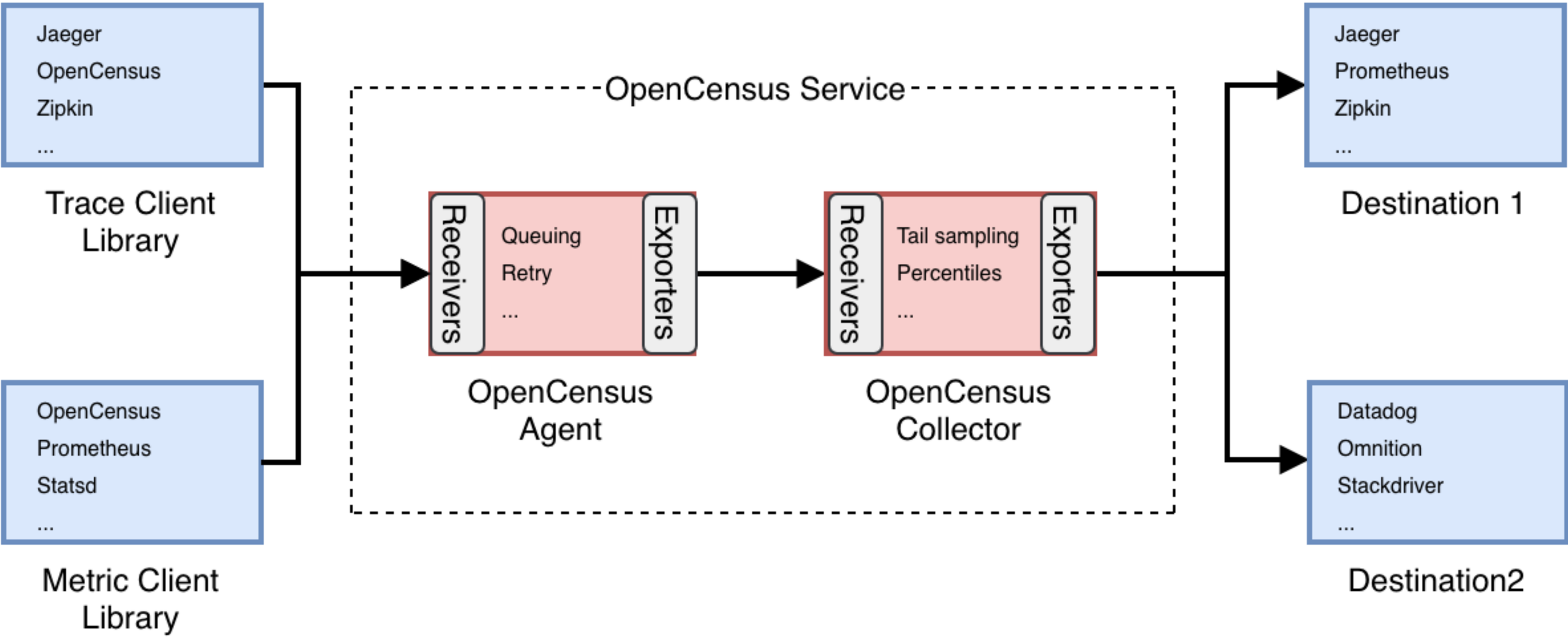
- Custom instrumentation
- Custom agent
- Custom translators

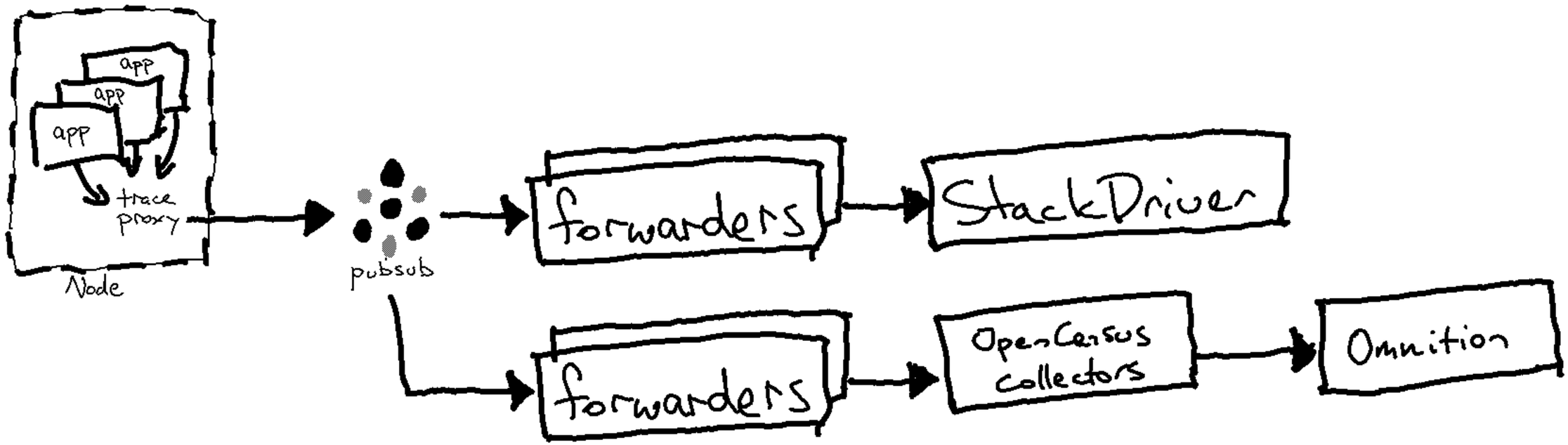
What’s wrong with this architecture?

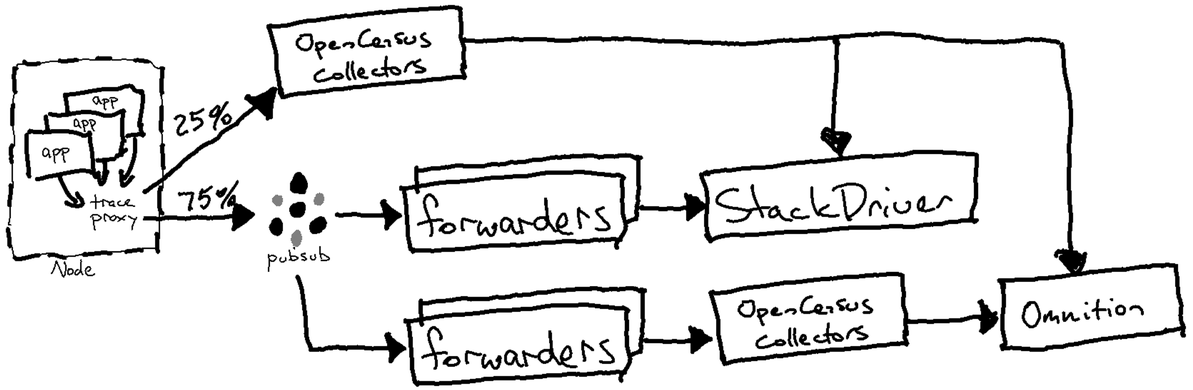
- CoreDNS? Nginx?
- Rust? Node? Python? Java?
- Unnecessary redundancy, impedance mismatch

➡️ Commoditization

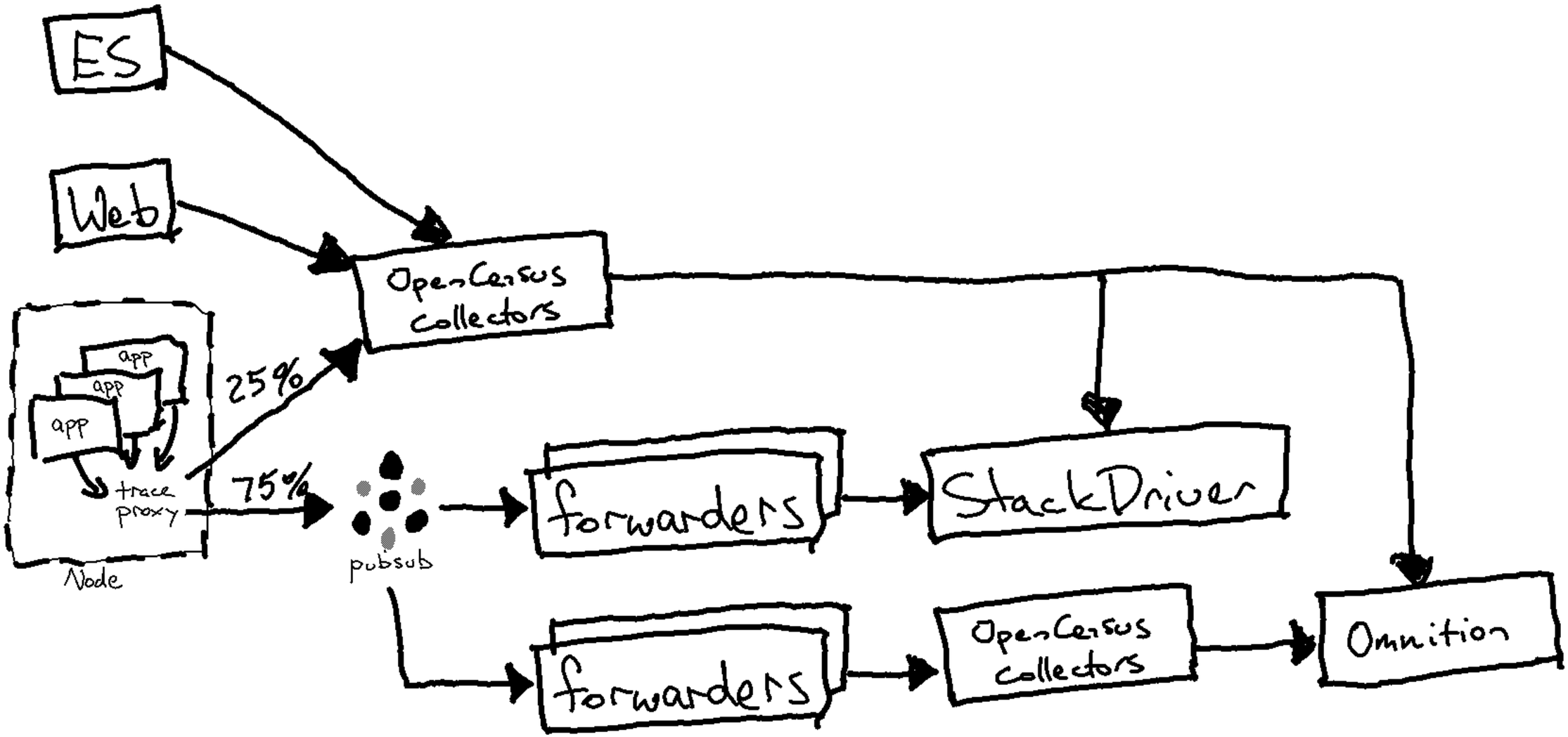
OpenCensus


Jaeger
OpenCensus
Zipkin
...
Trace Client Library
OpenCensus
Prometheus
Statsd
...
Metric Client Library
OpenCensus Service
Receivers
Queuing
Retry
...
Exporters
OpenCensus Agent
Receivers
Tail sampling
Percentiles
...
Exporters
OpenCensus Collector
Jaeger
Prometheus
Zipkin
...
Destination 1
Datadog
Omnition
Stackdriver
...
Destination2


app
app
app
trace
proxy
Node
pubsub
forwarders
StackDriver
forwarders
OpenCensus
collectors
Omnition




```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: trace-proxy-control
data:
  proxy-percent: "100"
...
        volumeMounts:
        - name: trace-proxy-control
          mountPath: /config/trace-proxy
      volumes:
      - name: trace-proxy-control
        configMap:
          name: trace-proxy-control
```

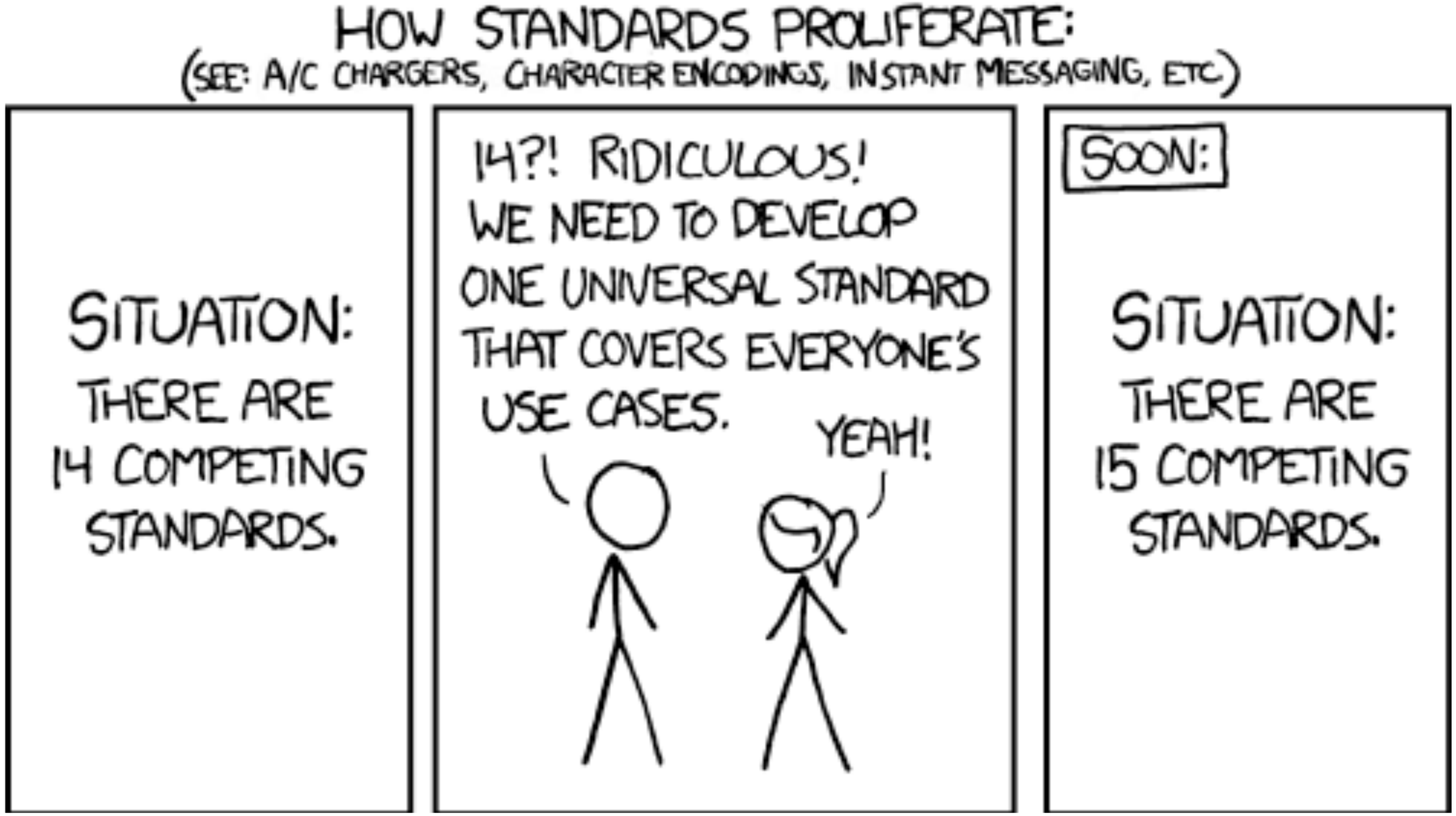

ES
Web
OpenCensus collectors
app
app
app
trace proxy
Node
25%
75%
pubsub
forwarders
StackDriver
forwarders
OpenCensus collectors
Omnition

What’s wrong with this architecture?

- ~~CoreDNS?, Nginx?~~
- ~~Node?, Java?~~
- ~~Unnecessary redundancy, impedance mismatch~~

➡️ Commoditization

OpenCensus + OpenTracing = OpenTelemetry



https://xkcd.com/927/

Steps toward OpenTelemetry

Steps toward OpenTelemetry

- OpenCensus collector => OpenTelemetry collector

Steps toward OpenTelemetry

- OpenCensus collector => OpenTelemetry collector
- trace-proxy => OpenTelemetry agent

```
pipelines:
  traces:
    receivers: [opencensus, jaeger, zipkin, shopify]
    processors:
    - memory_limiter
    - resource_labeler
    - batch
    - queued_retry
    exporters: [opencensus]
```

Steps toward OpenTelemetry

- OpenCensus collector => OpenTelemetry collector
- trace-proxy => OpenTelemetry agent
- OpenTelemetry Rust 🤷

Steps toward OpenTelemetry

- OpenCensus collector => OpenTelemetry collector
- trace-proxy => OpenTelemetry agent
- OpenTelemetry Rust 🤷
- OTLP exporter from custom instrumentation library

🌞 & 🌹?

- 😡 Agent troubles
- 😡 Breaking changes in collector, including metrics
- 😡 OTLP

What’s next for tracing at Shopify?

- OTLP in production
- Remove the agent?
- OpenTelemetry Ruby in production
- OpenTelemetry all the things

What’s next for tracing at Shopify?

- W3C traceparent
- Collector pool per region rather than per cluster
- Custom analysis backend using BigQuery

Lessons learned

- Migration takes a long time, especially in the early days
- Working backward from the end of the pipeline 💯
- Fine-grained traffic migration from old to new pipeline 💯
- Trace instrumentation and collection is commoditized

Thanks!

shopify