How to transparently migrate 300+ services to OpenTelemetry

Daniel Gomez Blanco
Principal Engineer @Skyscanner

OPENTRACING

OpenTelemetry

Who am I?

What am I doing here?

- Principal Engineer at Skyscanner
- We're trusted by 100M travellers/month
- Focus on observability and operational monitoring
- Reducing toil and cognitive load for the last 10 years
  - SKIPJAQ – ML-driven config optimisation
  - CERN – DB On Demand now running 1000+ DBs

Agenda

- Tracing at Skyscanner
- Motivation and constraints
- How we implemented OpenTelemetry
- Rolling out changes
- Hurdles and gotchas
- Next steps and OpenTracing deprecation
- Questions

Tracing at Skyscanner

A short history of migrations for 300+ Java, Python and Node services

AWS X-Ray on ECS

2018

OpenTracing + Lightstep

2019

Lightstep Satellites in K8s

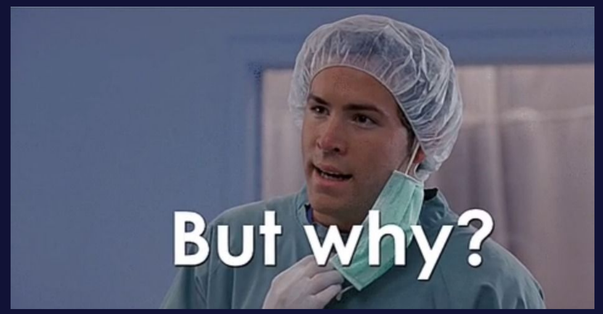
2020

OpenTracing upgrades and o11y strategy

2021

OpenTelemetry and beyond!

Our motivation for OpenTelemetry



- Our tracing abstraction was becoming a source of issues
  - Hard to keep up with API upgrades
  - Prone to memory leaks and orphaned spans
  - Context propagation sometimes faulty
- OpenTracing deprecation soon(-ish)
- Avoiding vendor lock-in
- Auto vs custom instrumentation
- Skyscanner-wide CNCF alignment

Constraints

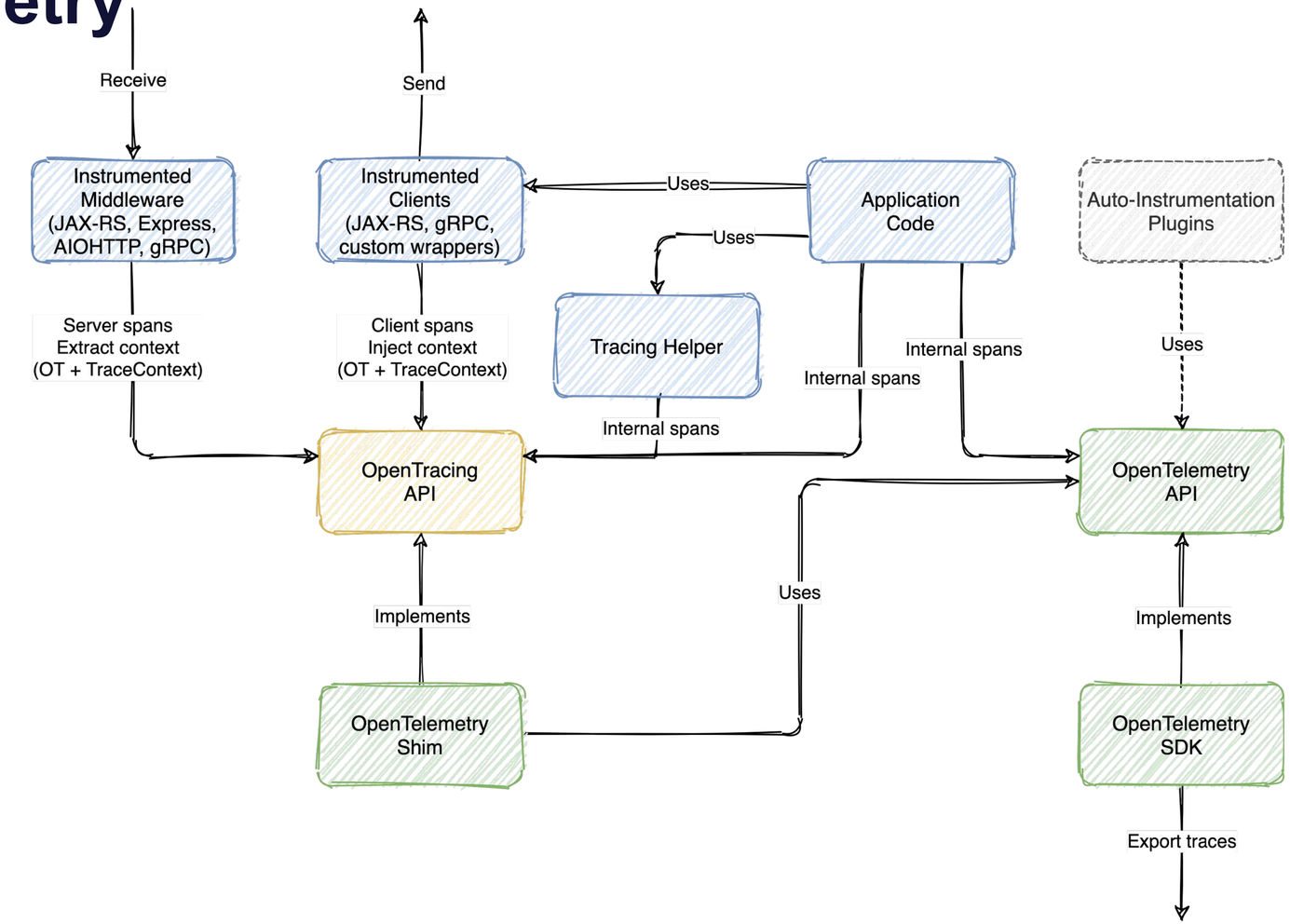
- No big asks from service owners
- No breaking changes in instrumentation
- No disjointed traces
- No orphaned spans
- No default attribute changes

How we implemented OpenTelemetry

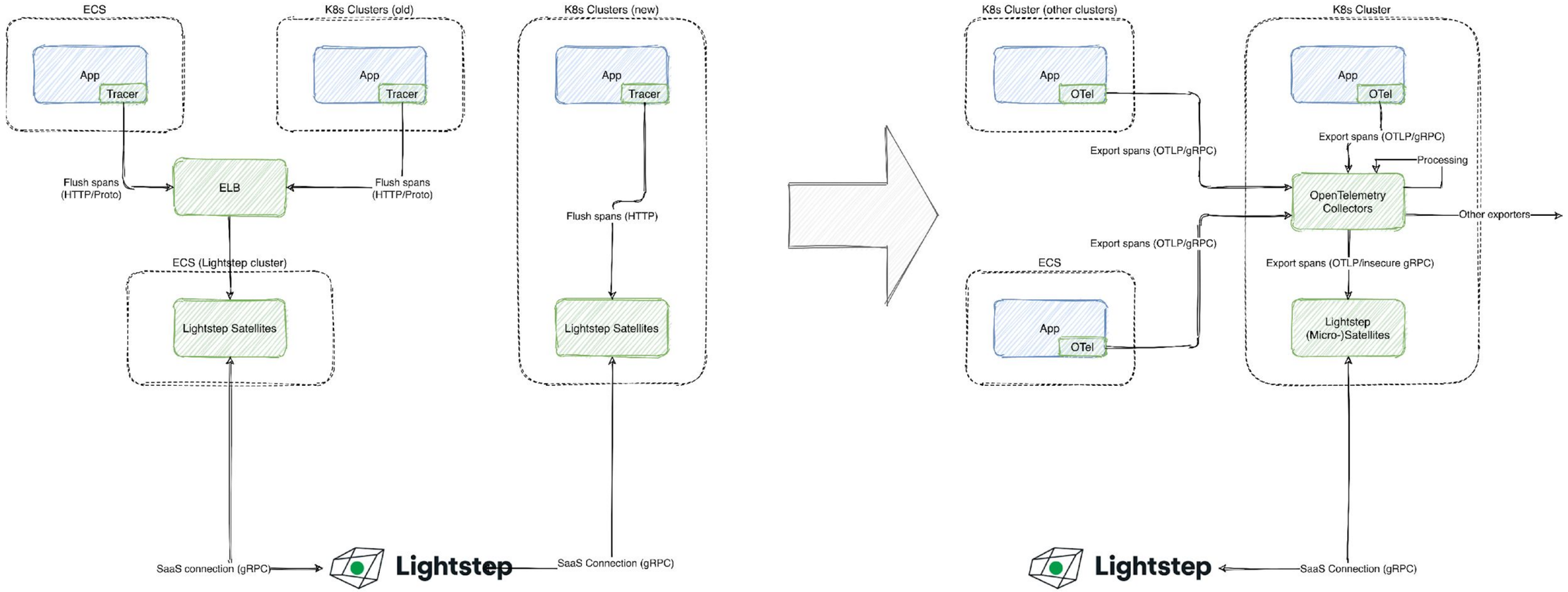
Our goal: to make the migration a version bump

- OpenTracing shims for backward compatible instrumentation
- Composite trace context propagators to avoid disjoined traces or orphaned spans
- OpenTelemetry Collectors to provide a cetralised export config
- **Common config libs (MShell)**
  - Configure SDK
  - Helper libraries
  - Apply resource attributes

Instrumentation with OpenTracing + OpenTelemetry



Collection & Export with OpenTracing + OpenTelemetry

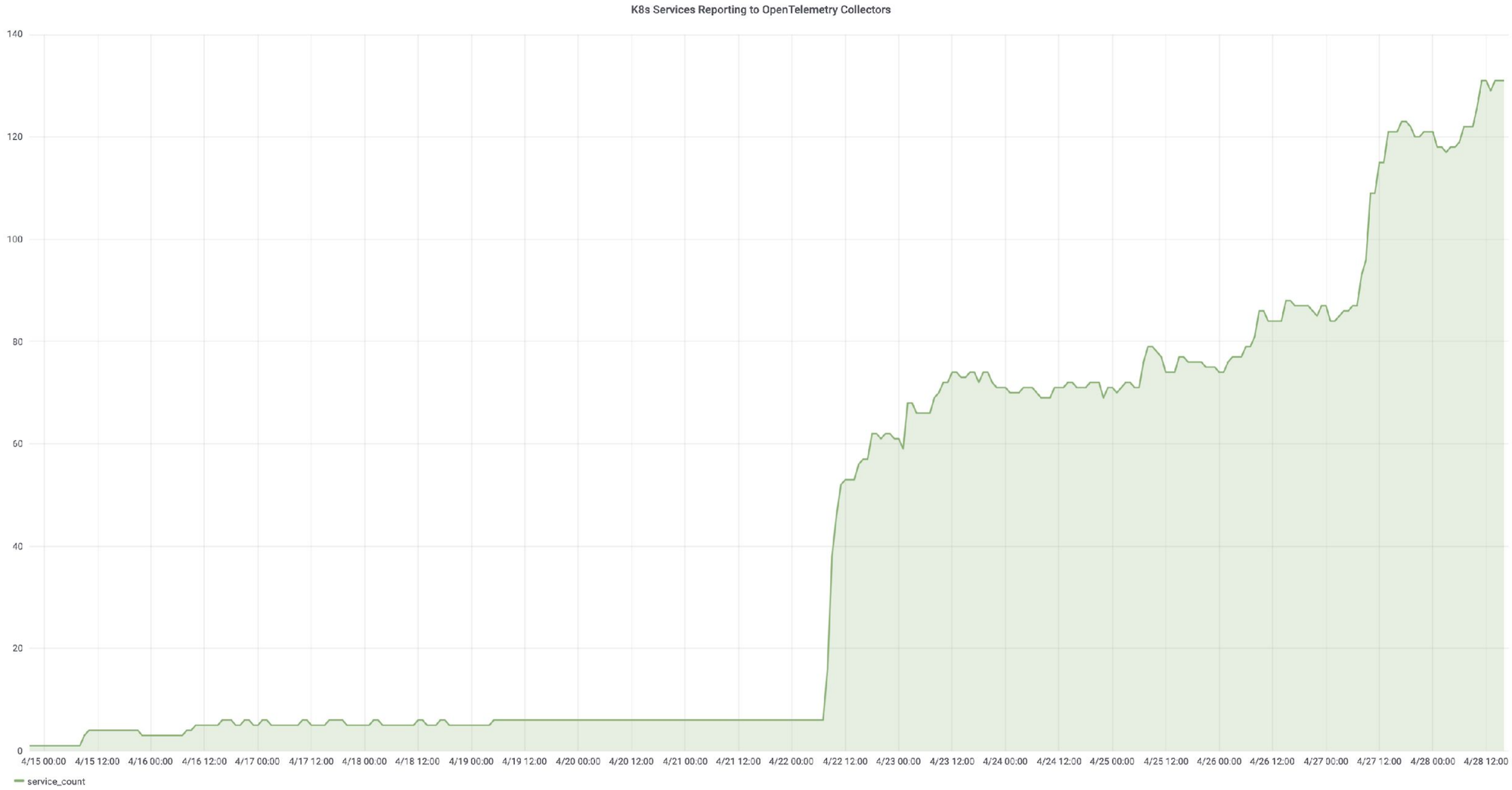


Rolling out changes

Our goal: to make the migration a version bump

- Tracing reference apps
  - Validate common scenarios
  - Showcase best practices
- Benchmarks in dev environments
- Early adopters behind feature flag
- Default to OpenTelemetry in our config libraries
- Dependabot version bump for service owners

OpenTelemetry Adoption Within the Mesh



Hurdles, gotchas...

… and other lessons we learnt along the way

- Trace ID truncation: upgrade your Lightstep Satellites!
- Order of composite propagators matters
- Never use SDK methods in plugins (we did and regretted it)
- OTel SDK more conservative on resources
- Tracer metrics need to be exported
- Global OTel instance re-registration is not a no-op
- SpanKind must be present at span creation

Next steps and OpenTracing deprecation

Our goal: to make the migration a version bump

- Slow deprecation of OpenTracing
- Adapting helper libs to OpenTelemetry
- Auto-instrumention to replace custom and OpenTracing plugins
- Adopting semantic conventions
- Reporting and budgeting
- Sampling at the Collector level
- Metrics and logs


Thank you, now time for questions!