Banked :
Instrumentation with OpenTelemetry

Contact
chris@banked.com
Chris Reeves — Golang Platform Engineer

Who am I?

Chris Reeves - Golang Platform Engineer

2014 - 2018
SOON_

2018 - 2019
Karhoo

2019
City Sprint (on the dot)

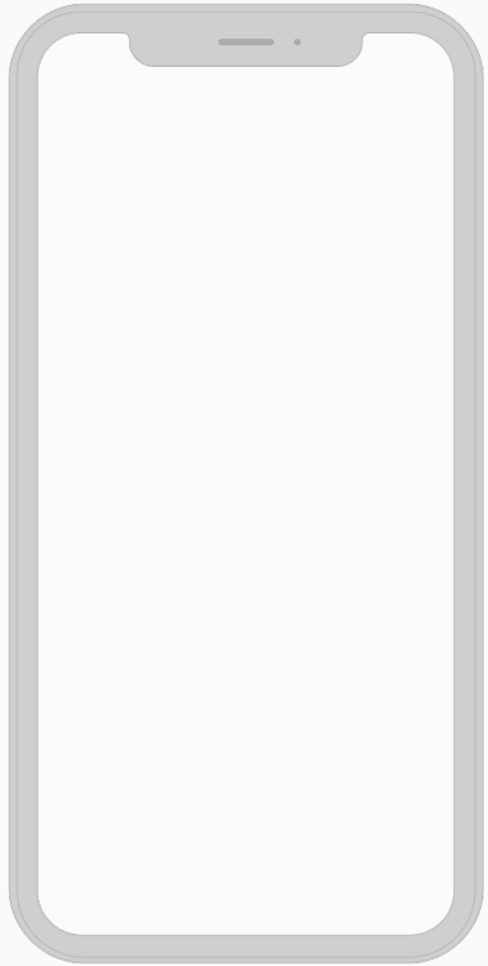
2019 — 2021
Vidsy

- Full time Go Engineer for 6 🤷‍♂️ years.
- @krak3n Twitter.
- @krak3n Github.
- linkedin.com/in/krak3n/

What is Banked?

A global payments network built on modern banking rails.

Banked powers real-time payments for consumers, businesses and banks, improving customer experience, payment security, business efficiency and cost effectiveness.



A better way to take and make payments.

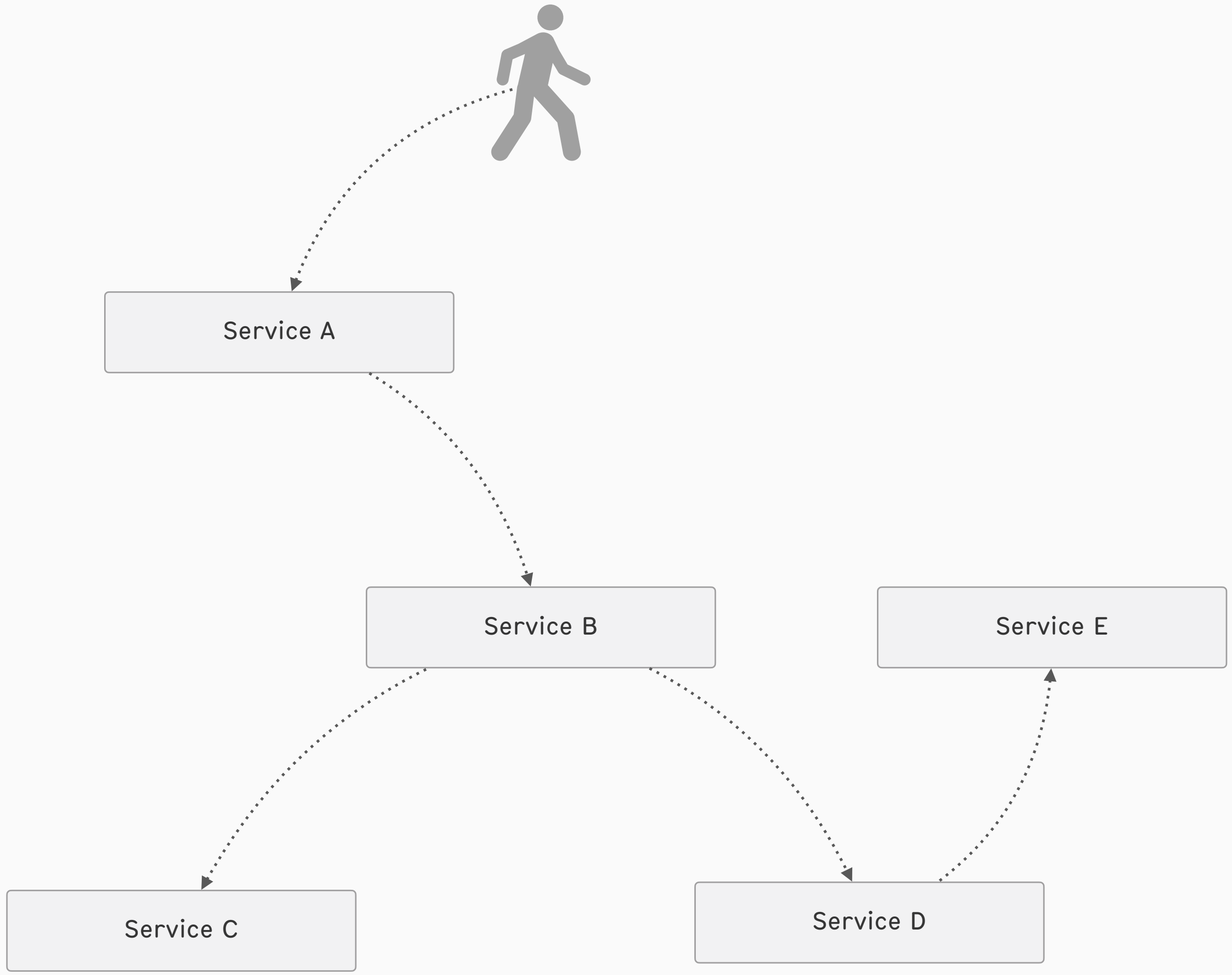
banked.com

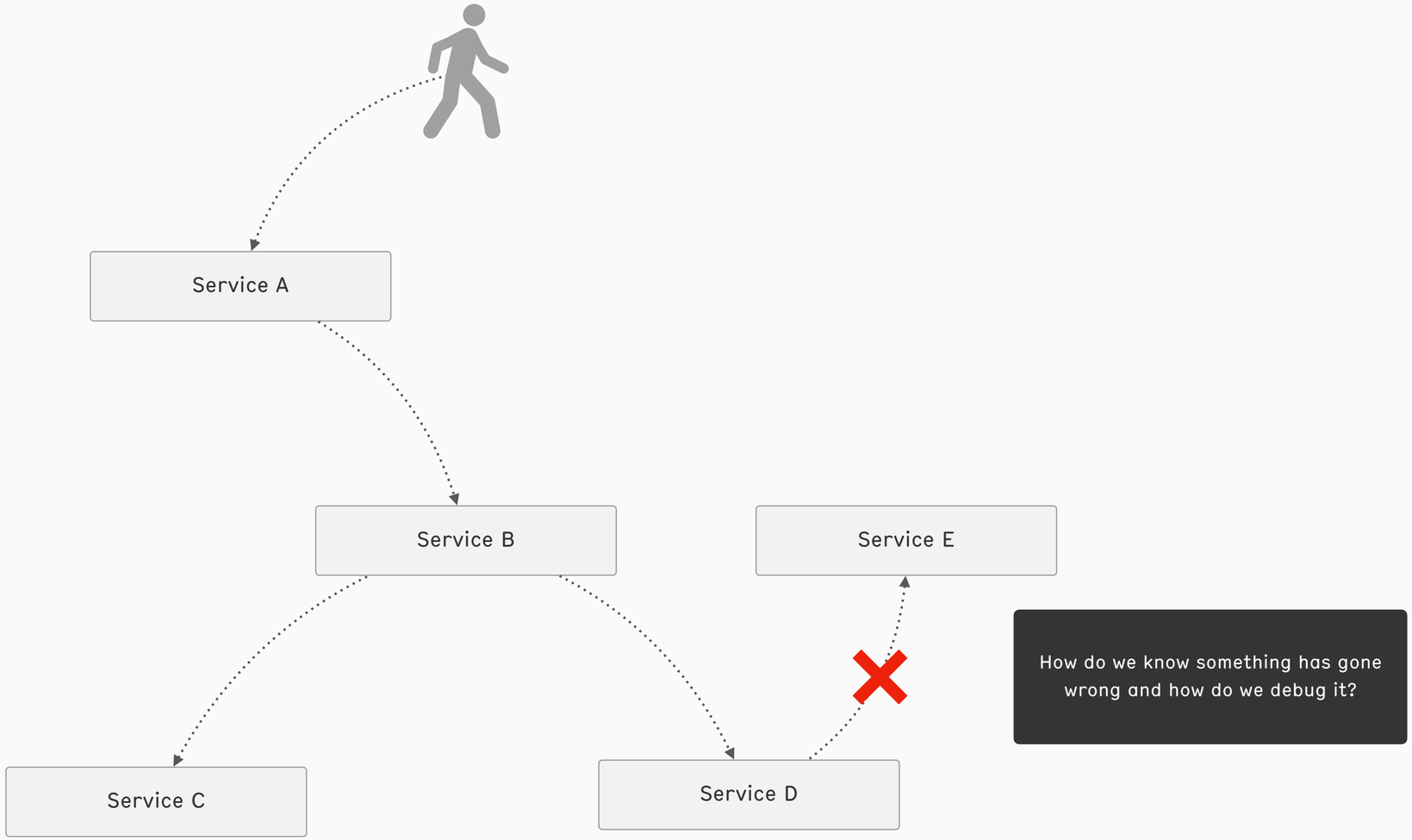
Open Banking.

“Open banking connects banks, third-parties and technical providers – enabling them to simply and securely exchange data to their customers’ benefit. We provide the trusted framework for collaboration, so you can deliver better value and create innovative services.”

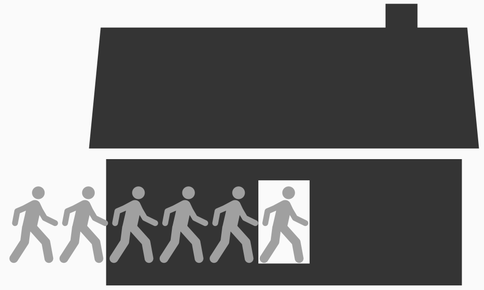
- 2015: The EU adopted the a revised Payment Services Directive to promote development / innovation of online & mobile payments through Open Banking.
- 2016: The UK CMA issued a ruling that the UK’s 9 biggest banks to allow licensed startups direct access to data / transactions.
- 2020: There are 202 FCA regulated providers who are enrolled in Open Banking.

Instrumentation.
What is it?




Service A
Service B
Service C
Service D
Service E
How do we know something has gone wrong and how do we debug it?

Three ~~pillars~~ windows of instrumentation.

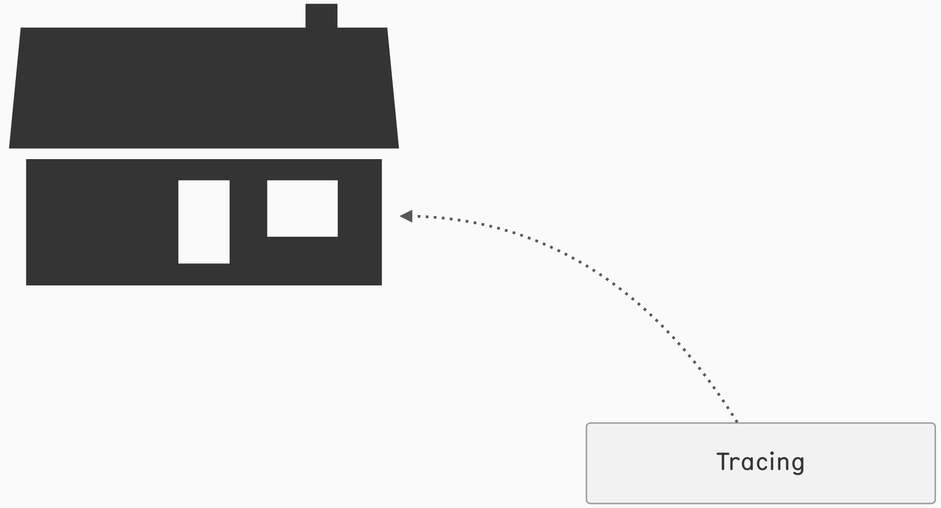


YOUR APPLICATION WITHOUT INSTRUMENTATION IS A WINDOWLESS HOUSE

Visibility

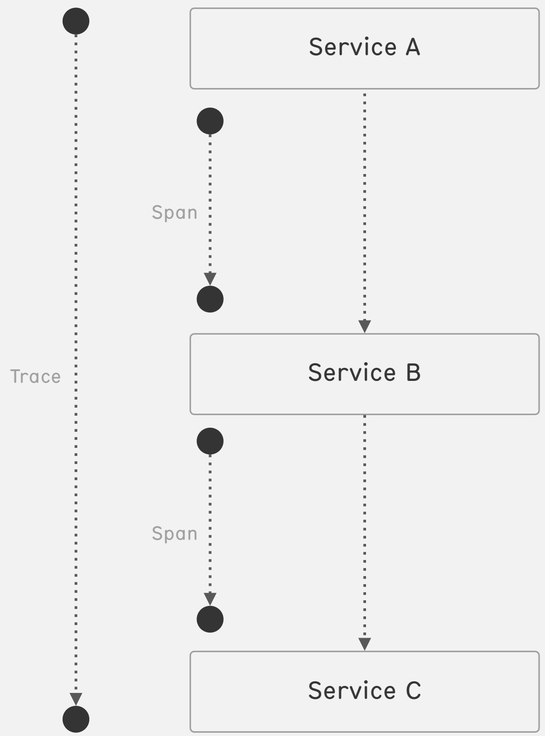
OBSERVABILITY

Three ~~pillars~~ windows of instrumentation.



Tracing.

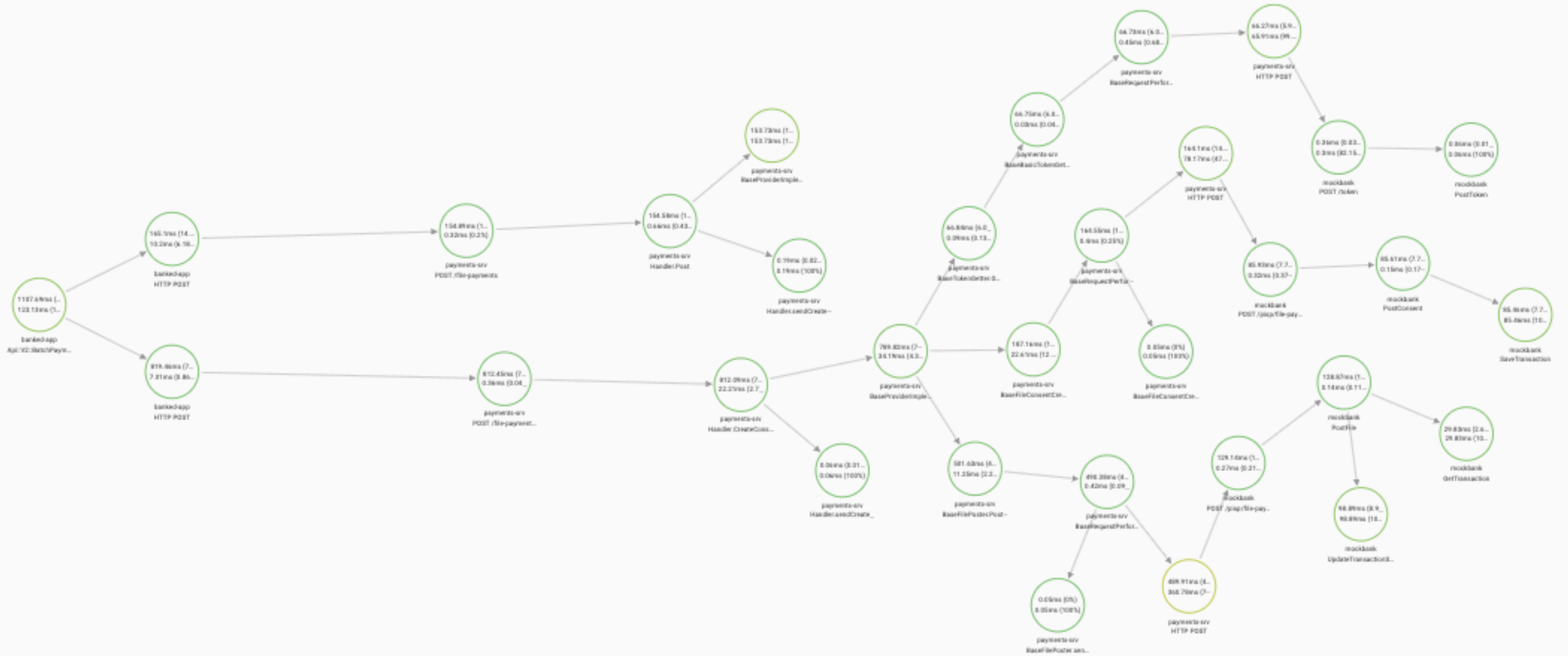
- Trace a single request through the platform
- Span code paths
- Reason large / complex systems
- Side effect visibility
- Discover performance bottlenecks



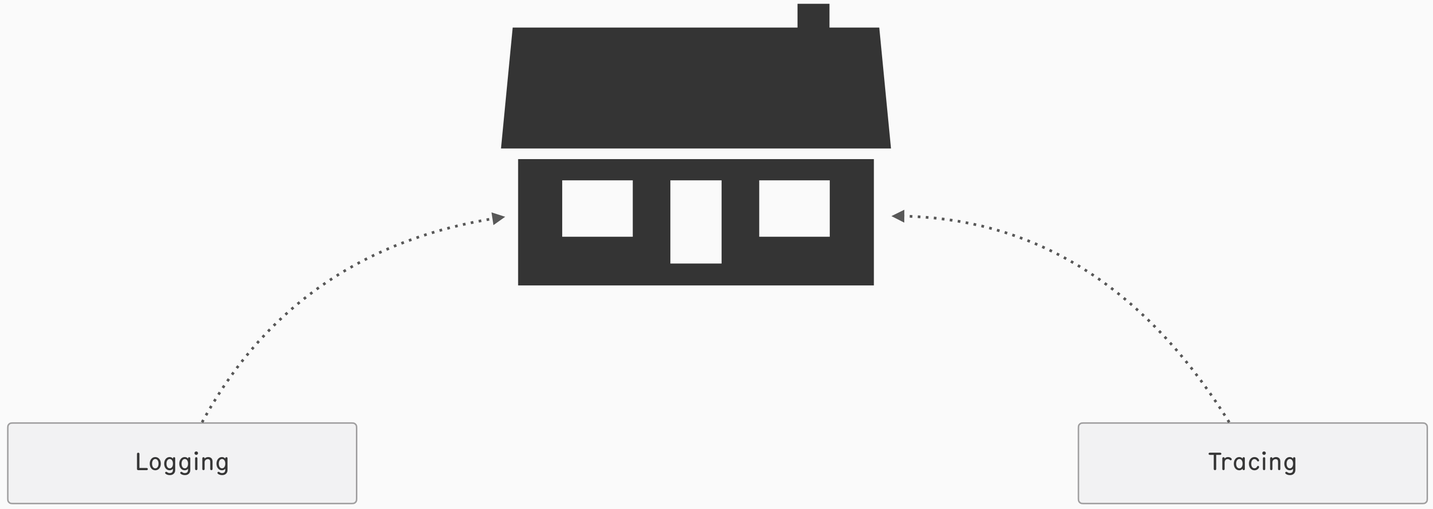

```go
package main

import "time"

func main() {
  span := StartSpan()

  // Do Some Work.

  ints := []int{1, 2, 3, 4}

  for _, i := range ints {
    time.Sleep(time.Second * i)
  }

  // Do Some More Work

  span.End()
}
```

Span

Trace Node Graph.



Three ~~pillars~~ windows of instrumentation.

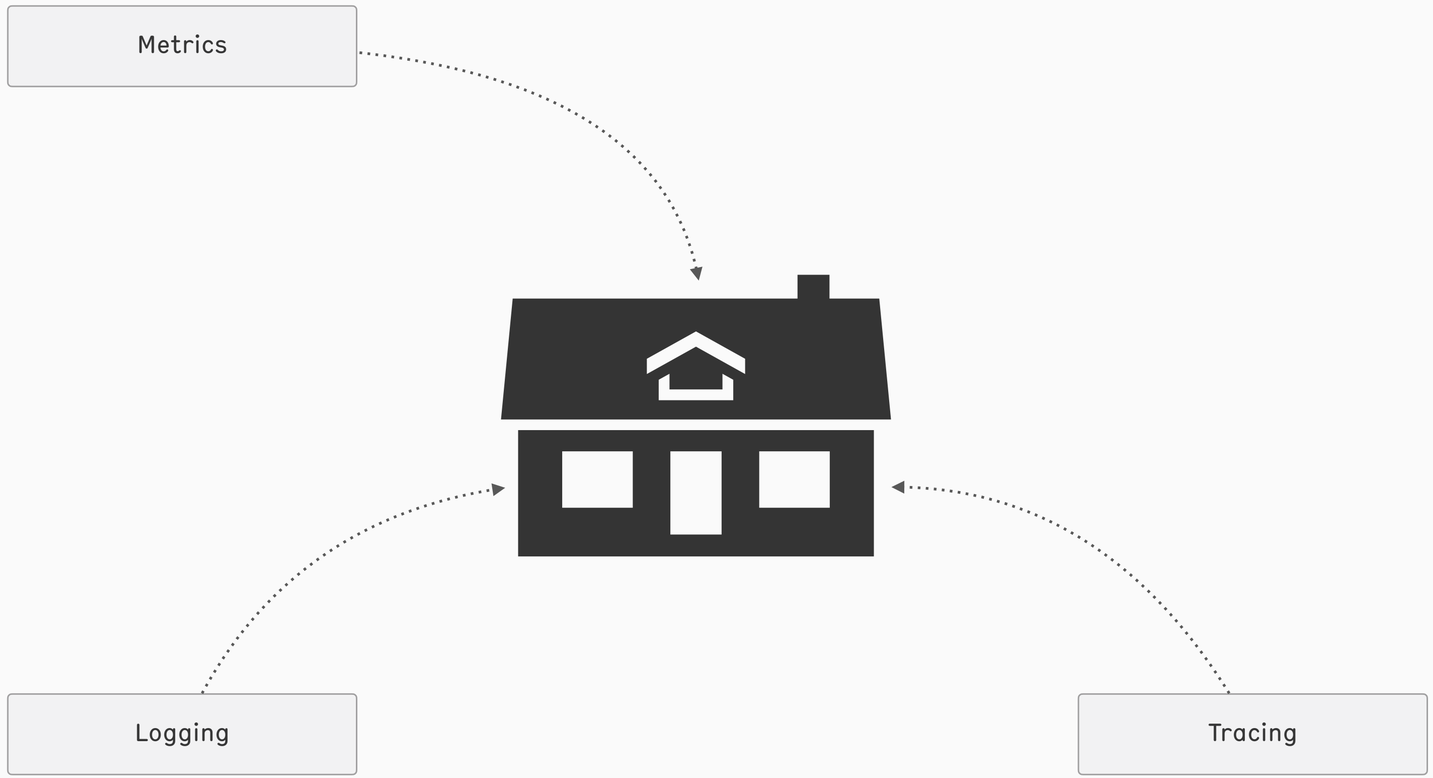


Logging.

- Gives us specific service information to aid in debugging
- They should be structured (e.g JSON formatted) for easy parsing
  - `github.com/rs/zerolog`
  - `github.com/uber-go/zap`
- They should provide enough context for engineers to be able to diagnose problems
- Include Trace/Span IDs so we can relate log lines back to their originating requests
- Consider including stack traces in errors e.g from `github.com/pkg/errors.WithStack`
- Propagate the logger on `context.Context` where possible

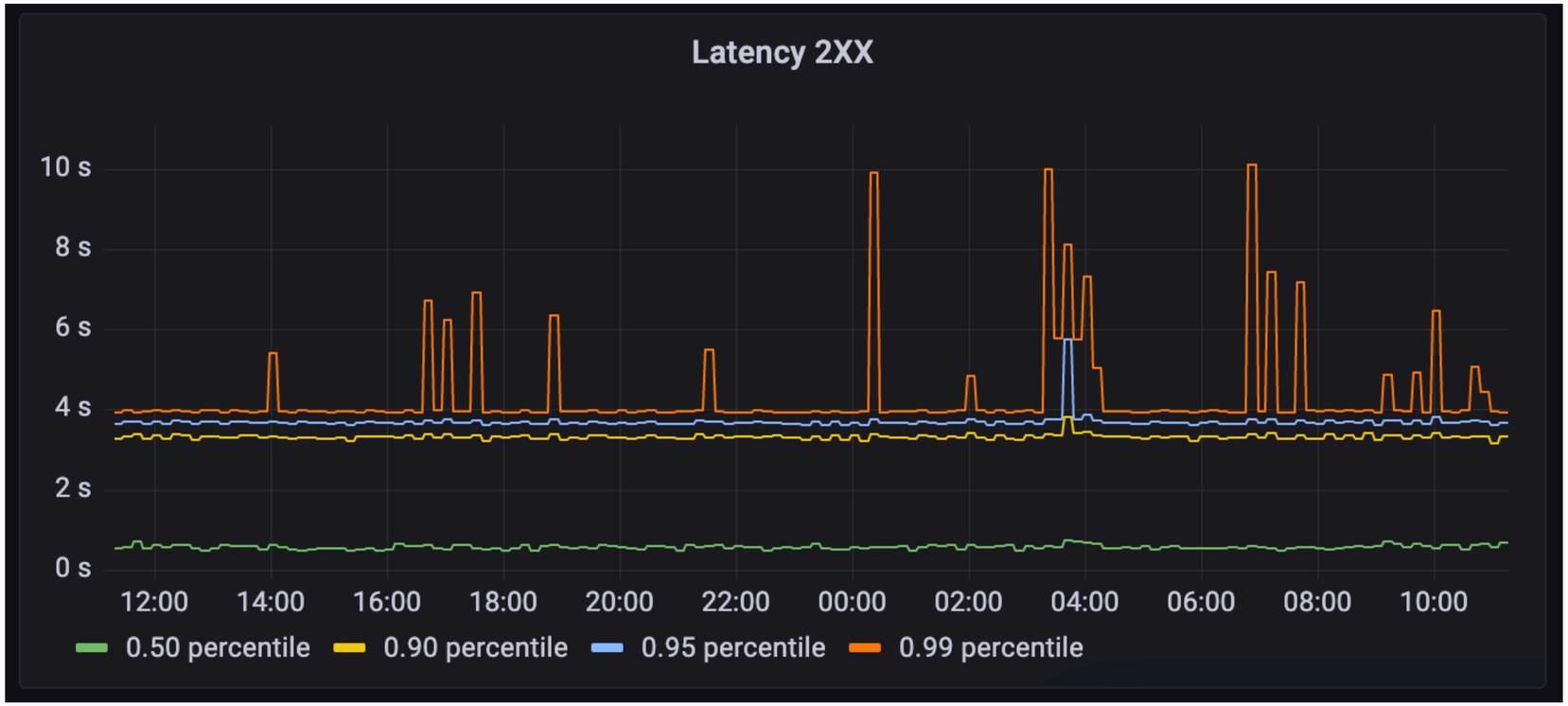
```
{
    "http": {
        "response": {
            "code": 200,
            "duration": 0.062182,
            "written": 0
        }
    },
    "level": "info",
    "message": "handled HTTP/1.1 request GET /",
    "service": “name-srv",
    "spanID": "7fc56dfff73550da",
    "timestamp": "2021-10-19T15:07:49Z",
    "traceID": "57fd1cb2b3d01ccb81acfbad31832b7c",
    "version": "2254bce5"
}
```

Three ~~pillars~~ windows of instrumentation.

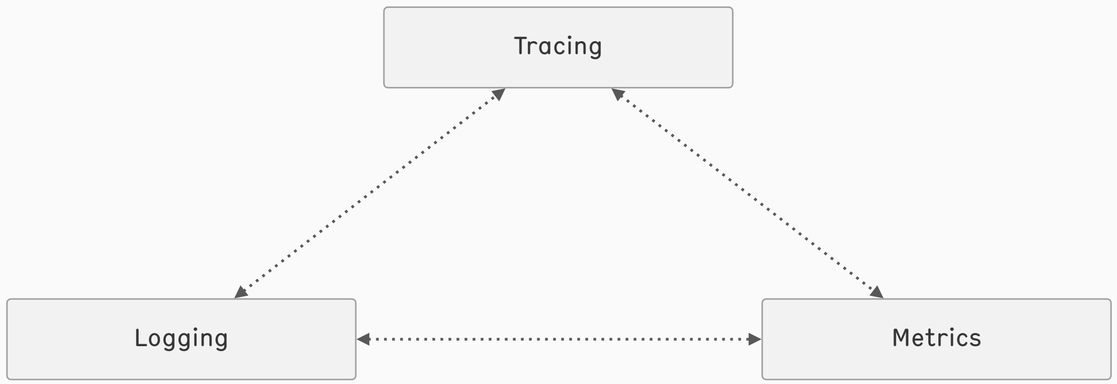


Metrics.

- Provide insights into application performance.
- Counters, Gauges (measurements) etc
  - Error counts
  - HTTP requests in/out counts
  - HTTP bytes in/out
- Alerting
- Business metrics
- Avoid log based metrics



Correlation.



APM

APPLICATION PERFORMANCE MONITORING/MANAGEMENT

Open Telemetry.

In the beginning

Vendor Lock In

Open(.*)

OPEN METRICS

OPEN TRACING

OPENCENSUS

OPEN TELEMETRY

Open Tracing / OpenCensus merge to form Open Telemetry in 2019

GITHUB.COM/OPEN-TELEMETRY

CNCF

Specification Driven

GITHUB.COM/OPEN-TELEMETRY/OPENTELEMETRY-SPECIFICATION

Trace Specification.

API.

- Stable / Feature Freeze
- Defines key concepts:
  - `TraceProvider`: Entry point to the API, provides `Tracer`'s.
  - `Tracer`: Responsible for creating `Span`'s.
  - `Span`: Is the API to trace an operation.

SDK.

- Stable
- Defines how the SDK should generally be designed and work across each language implementation
  - New `Tracer` instances should always be created via `TraceProvider`'s.
  - `TraceProvider`'s should implement `Shutdown` / `ForceFlush` methods.
  - Etc etc

Metrics Specification.

API.

- Feature Freeze
- Defines key concepts:
  - `MeterProvider`: Entry point to the API, provides `Meter`'s.
  - `Meter`: Responsible for returning `Instrument`'s.
  - `Instrument`: Responsible for taking a measurement (e.g a Counter).

SDK.

- Experimental
- Defines how the SDK should generally be designed and work across each language implementation
  - New `Meter` instances should always be created via `MeterProvider`'s.
  - `MeterProvider`'s should implement `Shutdown` / `ForceFlush` methods.
  - Etc etc

Logs Specification.

- Experimental
- Takes a different approach to Trace / Metrics (Clean Sheet Approach)
- Logs are one the biggest legacy challenges to solve.
  - Most languages either come with a built in logger or have well known / used 3rd party libraries
- For Open Telemetry to be successful in the logging space they would need to support the wide array logs / logging libraries while
- Existing logging libraries are weakly linked with observability.
- Solution: Open Telemetry Collector (we'll touch onto that later)

Notes about the Go SDK.

- There are two key libraries:
  - github.com/open-telemetry/opentelemetry-go: provides the core APIs for tracing/metrics etc
  - github.com/open-telemetry/opentelemetry-go-contrib: provides 3rd party instrumentation / exporters
    - Instrumentation for net/http
    - Exporters for Datadog
    - B3 / AWS Propagators
    - Shopify Sarama (Kafka library)
- Extensive use of sub-modules
- Trace SDK stable at 1.0
- Metric SDK < 1.0

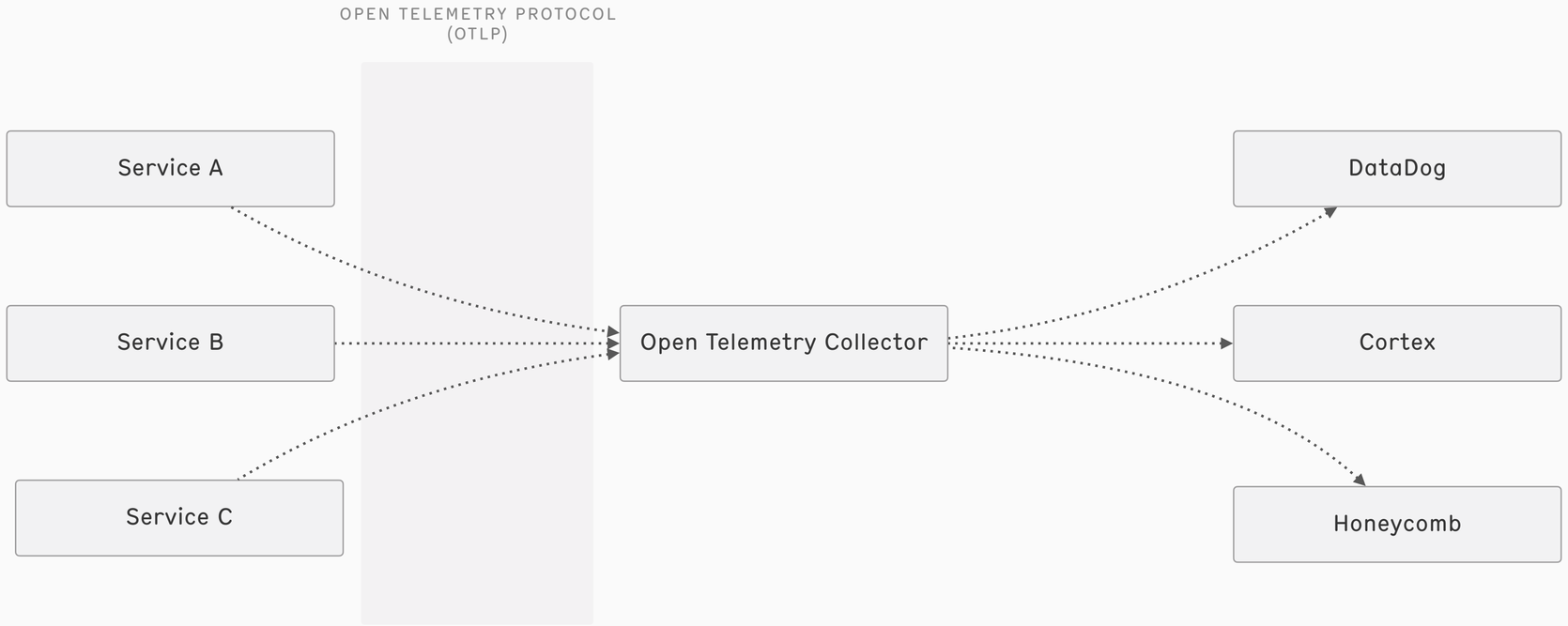
🙏 Demo Time

Code Examples:
https://gthub.com/banked/GopherConUK2021

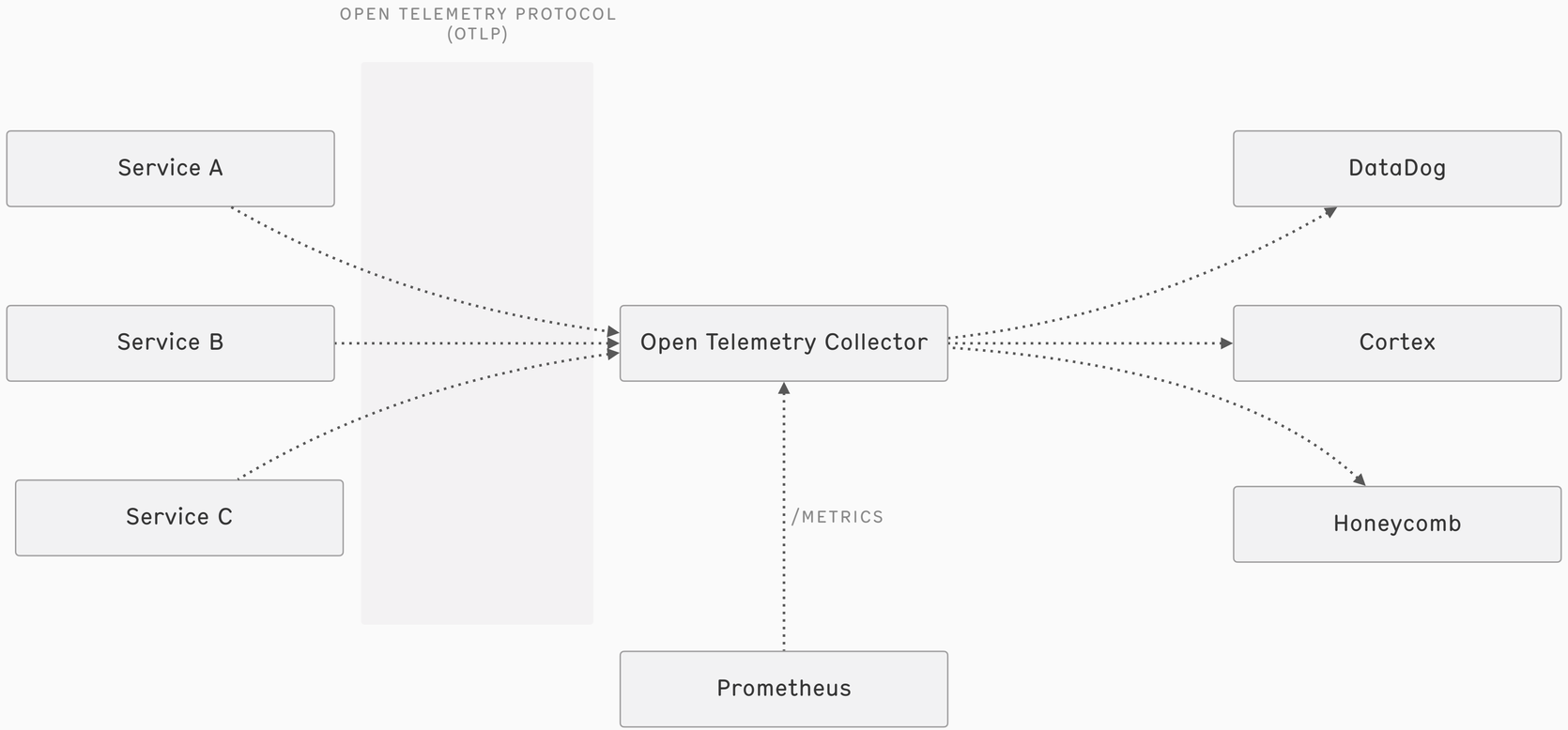
Open Telemetry Collector

GITHUB.COM/OPEN-TELEMETRY/OPENTELEMETRY-COLLECTOR

Open Telemetry Collector.



Open Telemetry Collector.
Prometheus.

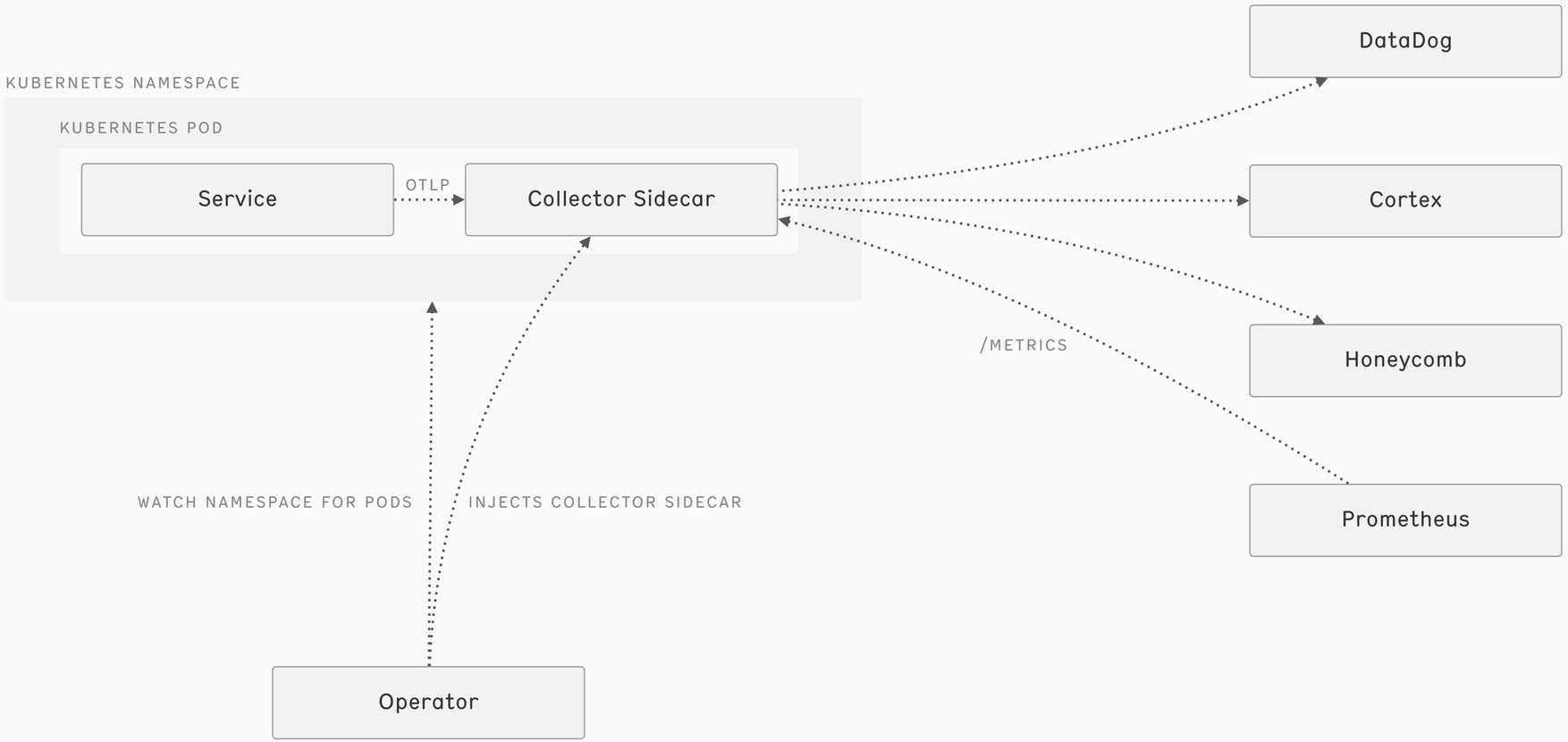



Otel Collector
OTLP
Jaeger
Prometheus
Receivers
Extensions: health, pprof, zpages
Batch
...
Attributes
Processors
Batch
...
Filter
Processors
Exporters
OTLP
Jaeger
Prometheus

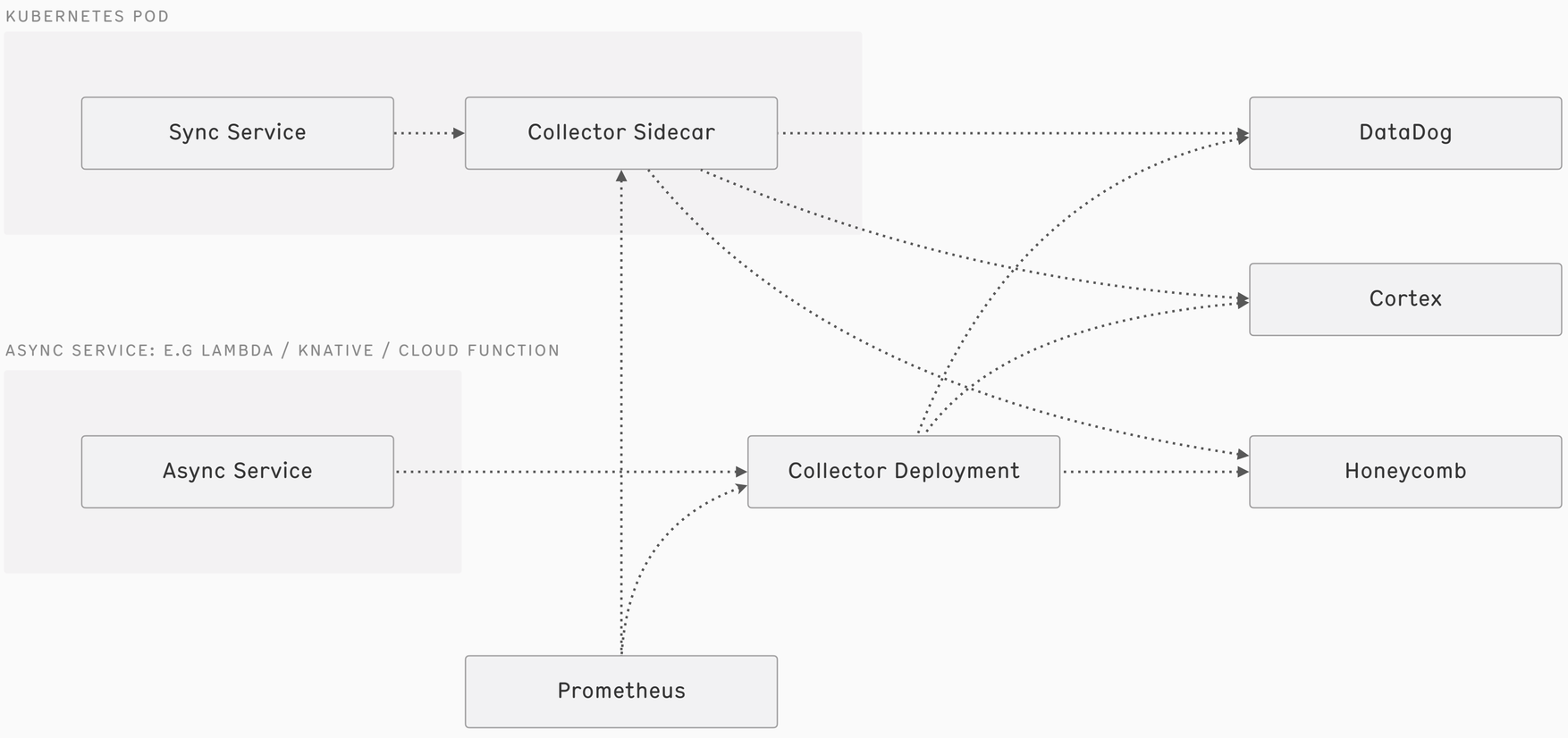
Open Telemetry Operator

GITHUB.COM/OPEN-TELEMETRY/OPENTELEMETRY-OPERATOR

Open Telemetry Collector.
Kubernetes Operator.

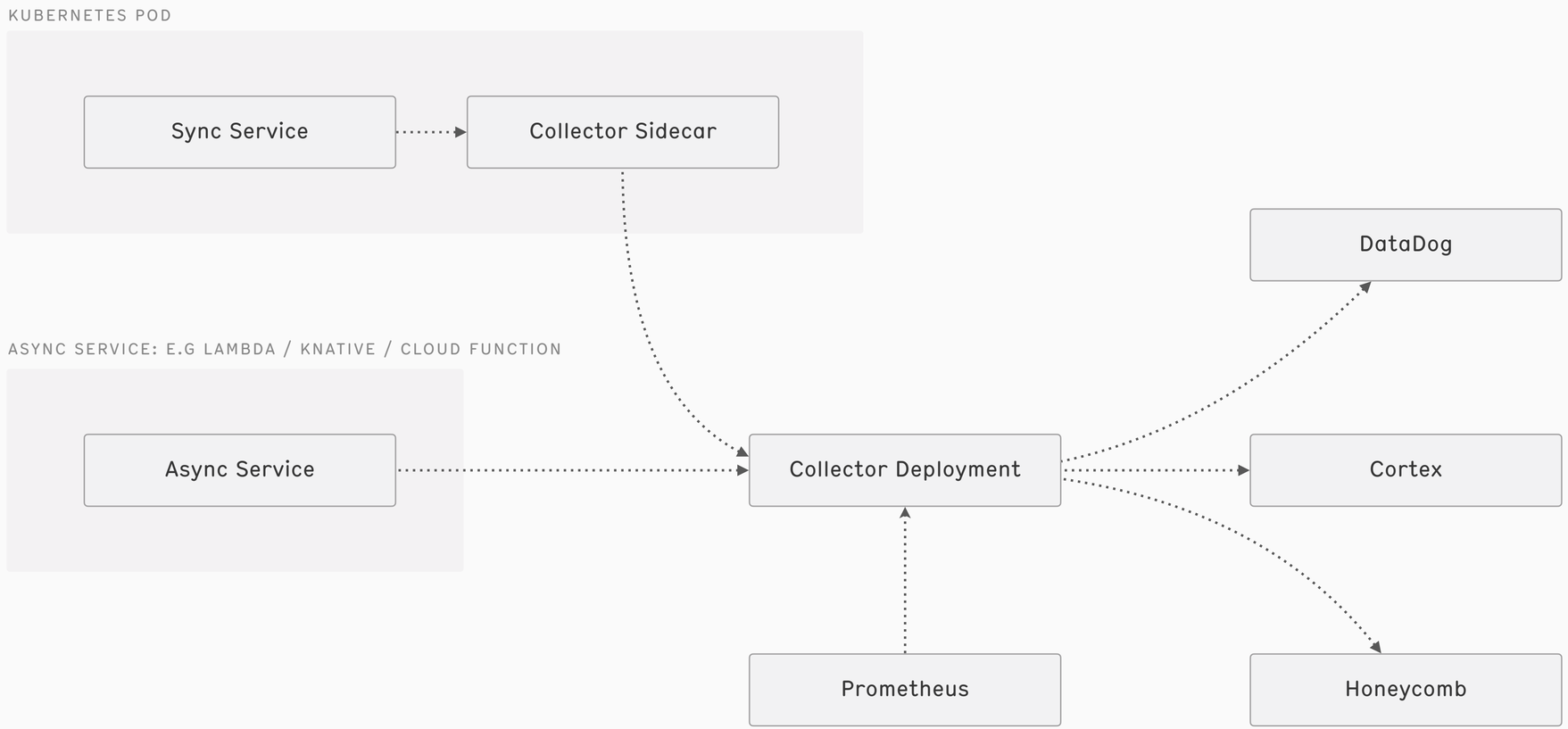


Open Telemetry Collector.
Sync / Async Option 1.



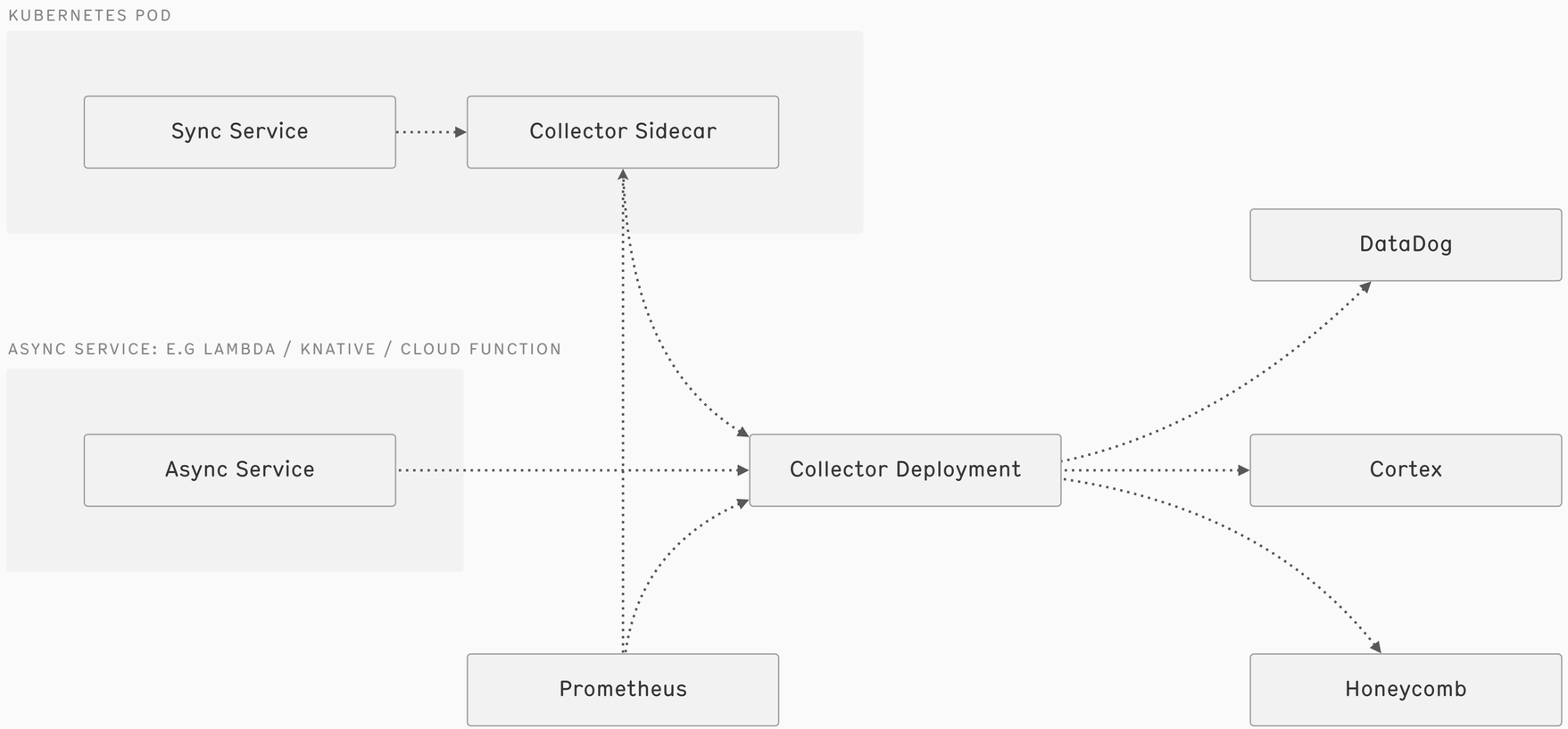
Open Telemetry Collector.

Sync / Async Option 2a.



Open Telemetry Collector.

Sync / Async Option 2b.



Wrapping Up.

We are hiring.

GITHUB.COM/BANKED/JOBS

Thank you :)

Contact
chris@banked.com
Chris Reeves — Golang Platform Engineer