M1 in Go, use Go library, test again

Test with metrics and logging only

Call M1 only if we have time

* What is Otel?
  + Collection of tools, APIs and SDKs
  + Instrument, generate, collect, export telemetry data
  + Successor of OpenTracing and OpenCensus – backwards compatible
  + Open Source!!
* Why Otel?
  + Standardized, compat with multiple services
  + Handles transferring of data to observability tools
  + Collector (will be covered soon)
* Tracing
  + Recording the lifetime of a call/request
  + Trace consists of tree of spans (stopwatches)
  + Span – stopwatch, metadata, context
  + Explain span lifecycle [https://opentelemetry.io/docs/concepts/data-sources/#traces](https://opentelemetry.io/docs/concepts/data-sources/%23traces)
* Metrics
  + Data recorded at regular intervals
* Logs
  + Timestamped data
  + With Otel, logs contain Trace IDs which can be used to inspect a given trace
* Collector
  + Connect tools which comm over different protocols (OTLP handles)
  + Vendor-agnostic way to receive, process and export telemetry data.
  + Processors can sample, filter, batch, refine information (renaming, add metadata)
  + Use image from <https://opentelemetry.io/docs/collector/>
  + Contrast with contrib
* Instrumentation
  + Auto vs Manual
  + Auto handles all steps of creation + sending of spans for you
  + With libs this can extend to propagation
  + Manual – more control over data/metadata as well as starting/stopping
  + Manual – add events, tags/attributes, control which funcs create spans
  + Depending on lang these can be used in conjunction or mutually exclusive
* System Overview
  + System Diagram
  + Explain broad goals and functionalities
  + Show example(s) of system traces
  + Cover technologies used (jaeger, etc)
* .Net Instrumentation
  + Show code snippet of program.cs, explain setup (ASP.NET for RESTful API)
  + Service name: everything before last . is domain, after = service
  + Walk through m1 controller
  + Helper func – start new span
  + M2 – Show consumer startup. Mongo auto instrumentation, sources needed. Manual prop in rabbit. Walk through extracting/injecting context.
* Collector config
  + Show docker compose first and show ports exposed, mount config files etc
  + Explain loadbalancer
  + Walk through actual collector
  + Discuss potential deprecation of tail sampler
* Load Testing
  + Discuss k6
  + Discuss overheads/results
* Tools
  + Live demo jaeger, prom, logs, elastic
* What does the future hold?
  + Potential deprecation tail sampling
  + Nothing unstable at this point
  + Survey – companies building their own distrib tracing systems, etc
* Misc
  + Demo <https://openapm.io/landscape>
* Q&A