Observability Mythbusters: Yes, You CAN Use Span Links in Lightstep

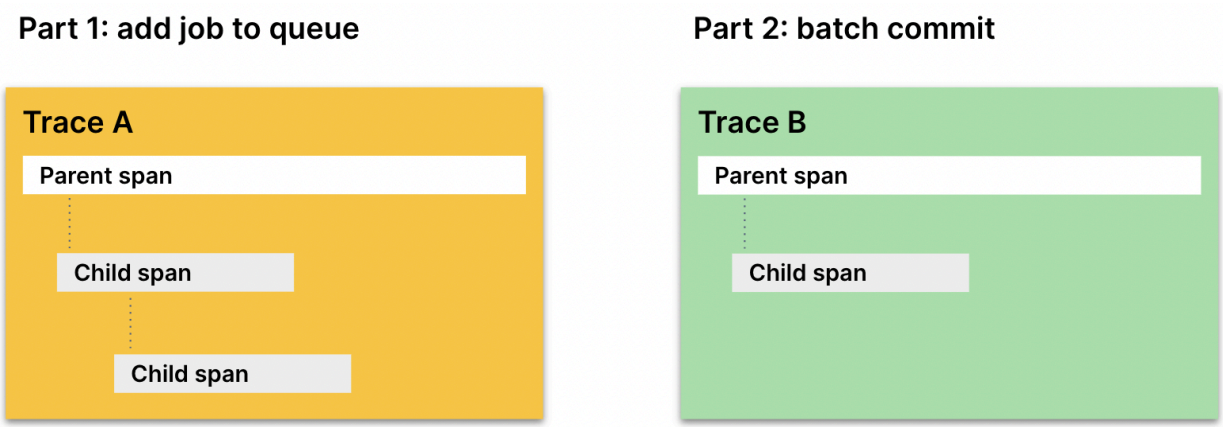
One of the things that I've enjoyed as part of my personal Observability Journey has been learning about different aspects of OpenTelemetry. This week, I got to dig into a concept that was totally new to me: Span Links. I have to admit that this one was a tricky one to wrap my head around, because the definitions out there are um…confusing. So I made it my mission to better understand Span Links, so that I can tell y'all what they're all about, and how they relate to Lightstep Observability. Are y'all ready for this? Let's do it!

A Tale of Two Services

To better understand Span Links, let's start with an example.

Suppose that you have an event-driven application. As part of this application, you have a service that **adds a job to a queue**. The jobs being added to the queue are executed asynchronously. One such job is a **batch commit** job. The "add job to queue" service and the "batch commit" service are each represented by a Trace.

Here's a handy little visualization, for your convenience.



We know that the "add job to queue" and the "batch commit" services are related, since the "batch commit" service was queued up by the "add job to queue" service. Unfortunately, we are faced with a problem.

Our "batch commit" service runs asynchronously. This means that when the "add job to queue" service queues up "batch commit", the work of "add job to queue" is done. This means that "add

job to queue" service has *no frickin' clue* as to when the "batch commit" service actually *starts*. Well if that's the case, how could you even *link* the two??

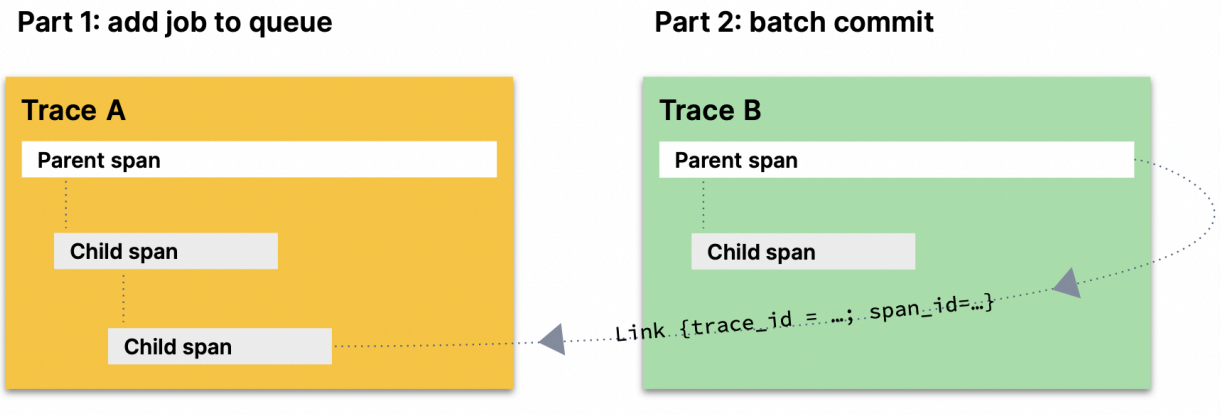
To add insult to injury, we've started seeing some performance issues in our app. (Cue panic!) And although we are getting some insights about our services from the individual Traces, we have a missing link, if you will, and without it, we don't have all the visibility into what's going on with our system at a higher level.

Okay, we know that in principle, there's a connection between these two Traces! Which begs the question: *is there a mechanism by which we can link these two Traces together?* Um…Is the sky blue? It sure is. Are llamas awesome? Yes, yes they are. Does OpenTelemetry have our backs? Most *definitely* yes!

And so, yes, we can link our above Traces, thanks to a lovely OpenTelemetry concept called a Span Link. A Span Link allows us to create a relationship between two Spans that are causally related. A causal relationship is one whereby one event causes another. In our case, the "add job to queue" service spawns a "batch commit" service, which will run at some point in time.

Or, in super duper layman's terms, when you have asynchronous processing, a Span Link gives the impression that the asynchronous process happened synchronously.

So with Span Links, our two services now look like this:



Yay – they're connected! Thank you Span Links!

Span Links & Observability Back-Ends

Okay…so Span Links come to the rescue, and all is well with the world of Observability and OpenTelemetry…amirite? Well…kind of?

So here's the rub. While Span Links *are* a thing in OpenTelemetry, not all Observability vendors support them. This means that even if you instrument your code to use Span Links, if the vendor back-end has no way to render them, then it's pretty much the same as not using Span Links at all, because you lose out on that added bit of visibility that Span Links give you. Kinda sucks, doesn't it? For example, at the time of this writing, in spite of requests from users, vendors such as NewRelic (see here) and Datadog (see here, here, and here) don't support Span Links. I did a little quickie search on Span Links for Splunk, and couldn't find any mention of Span Links.

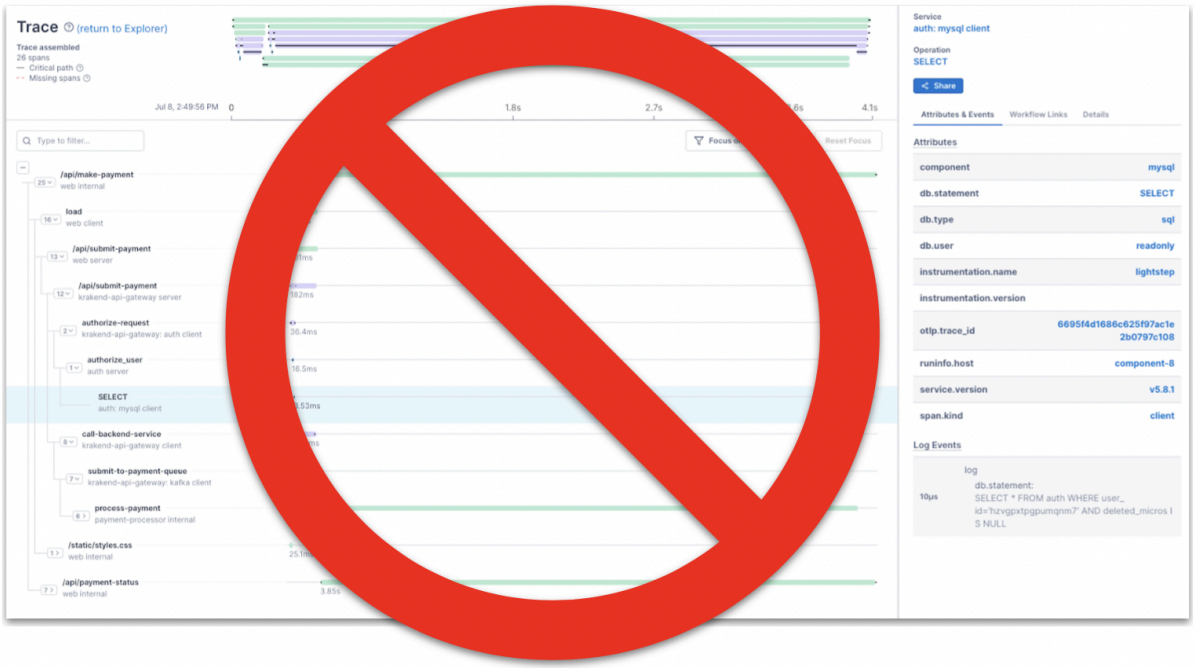
Luckily, there are a few Observability vendors that *do* support Span Links, including Honeycomb and Grafana. And now, as of this writing, you can also add Lightstep to the mix!

So…why now? Uh…why *not* now? As co-creators and active contributors to OpenTelemetry, we're "all in" on OpenTelemetry, and we want our product to reflect that by supporting OpenTelemetry features like Span Links. We also know that this is a feature that's important to our customers, and we want to say that we're listening.

Image source here

Span Links & Lightstep

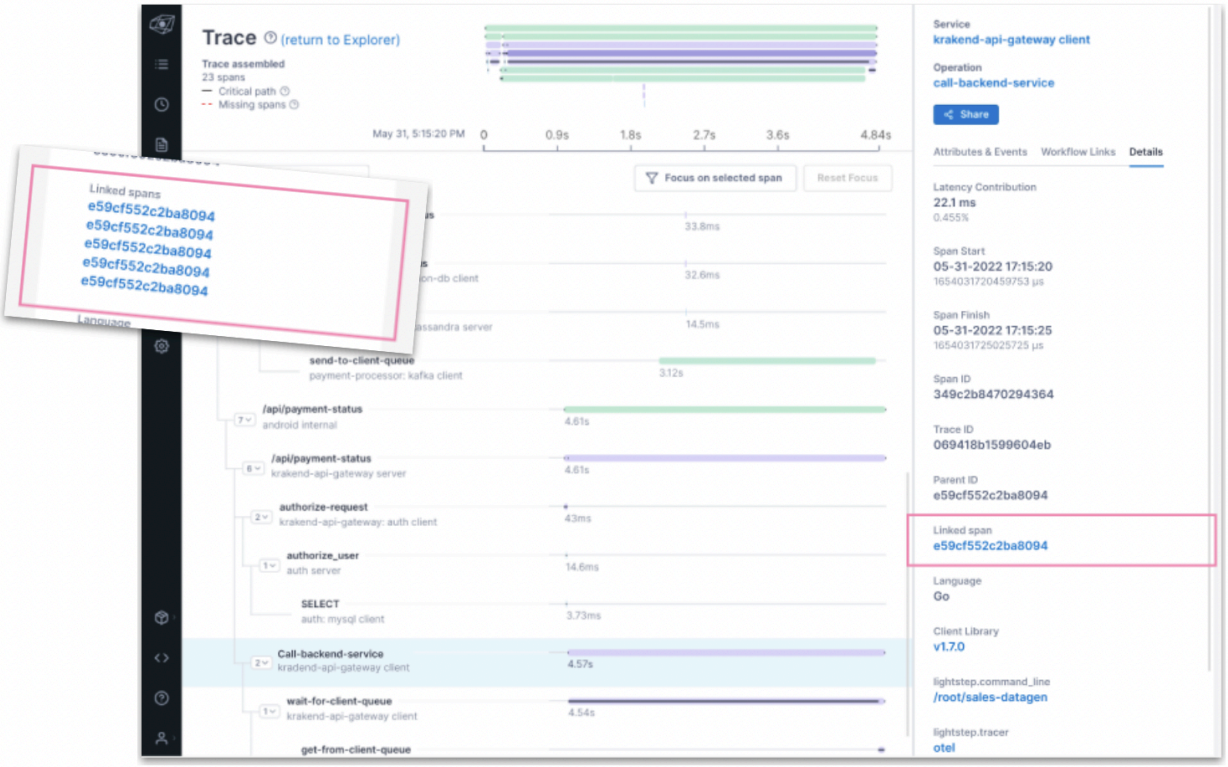
Okay, so let's see what Span Links look like in Lightstep. In the Before Times, Lightstep had no way of surfacing Span Links at all. Which looked something like this:



Booo! This meant that, going back to the earlier example with our “add job to queue” and “batch commit” services, we:

- Had no way to understand end-to-end performance of a particular request
- In looking at a slow batch commit, we didn’t know which writes were affected

Fortunately, that’s no longer the case!! Now, with Span Links enabled in our UI, it means that we now have this:



Rejoice, for we have Span Links!

Final Thoughts

As we've learned today, Span Links are pretty handy, because they give us the ability to show the relationship between causally-related processes. This is especially handy when it comes to event-driven architectures, allowing us to relate one or more asynchronous services with the service that spawned them.

Lightstep now supports Span Links in its UI, which means that you now have a way to visualize and troubleshoot these causal relationships, giving you more power and insight to help you answer the question, "Why is this happening?"

We invite you to give Span Links in Lightstep a try to see for yourself, and tell us what you think. Connect with us on the Lightstep Community Discord, or get in touch by e-mail. Hope to hear from y'all!

Now, please enjoy this picture of an alpaca.

Photo by Joakim Honkasalo on Unsplash

Peace, love, and code. 🦄🌈💫