

TIL: Mermaid Gantt diagrams are great for displaying distributed traces in Markdown

2023 / 03 / 31

Tagged [til](#) [jekyll](#) [mermaidjs](#) [opentelemetry](#) [tracing](#)

Today I noticed via [a tweet](#) by [@mitsuhiko](#) that [Mermaid](#) Gantt diagrams are great for displaying distributed trace information like what you'd get from [JaegerUI](#). I've been working with [OpenTelemetry](#) a fair bit recently and, in recent projects, I've been including screenshots of JaegerUI whenever I need to show a distributed trace in my documentation. This generally works fine but I'm happy to have an alternative that's more at home in Markdown and on the web.

If you're not familiar with Mermaid, they have [great docs](#).

Gantt Diagrams

Gantt diagrams are typically used for scheduling multiple tasks along a shared timeline. In hindsight, it makes total sense to reach for a Gantt diagrams for diagraming a distributed trace.

The Mermaid syntax for a pretty typical Gantt looks like:

```
gantt
  title A Gantt Diagram
  dateFormat YYYY-MM-DD
  section Section
  A task           : a1, 2014-01-01, 30d
  Another task     : after a1 , 20d
  section Another
  Task in sec     : 2014-01-12 , 12d
  another task    : 24d
```

and, when rendered, looks like:

A Basic Trace Diagram

The [tweet](#) I mentioned previously shows code for a Gantt diagrams of a simple trace:

```
gantt title Trace Showing Attached and Detached Spans dateFormat x axisFormat %S.%L
section Frontend /checkout :crit, 0, 500ms App :300, 180ms POST /api/analytics :done,
450, 70ms GET /assistant/poll :done, 450, 120ms POST /api/analytics :done, 580, 70ms
```

The code for which is:

```
gantt
  title Trace Showing Attached and Detached Spans
  dateFormat x
  axisFormat %S.%L

  section Frontend
  /checkout          :crit, 0, 500ms
  App                :300, 180ms
  POST /api/analytics :done, 450, 70ms
  GET /assistant/poll :done, 450, 120ms
  POST /api/analytics :done, 580, 70ms
```

To do this, the code above uses just a few features of Mermaid's [Gantt syntax](#) to make the diagram look less like a typical Gantt diagrams and more like an OpenTelemetry trace:

1. To show everything on a time scale instead of a calendar date scale:
 - Specify a dateFormat of x (milliseconds) instead of the usual YYYY-MM-DD
 - Specify an axisFormat of %S.%L which makes the chart use seconds with milliseconds instead of dates
2. Separate each service into its own section
3. Visually distinguish spans using tags like :crit, :done which apply styling by default

A More Realistic Example

[@mitsuhiko](#) also linked to a [Sentry RFC](#) that's in the works with a more representative example:

```
gantt title Example Starfish Trace dateFormat x axisFormat %S.%L section Frontend
/checkout :crit, 0, 1500ms GET /api/session :150, 170ms POST /api/analytics :190, 70ms
GET /api/checkout/state :200, 500ms GET /api/checkout/cart :1100, 140ms App :1300,
180ms POST /api/analytics :done, 1450, 70ms GET /assistant/poll :done, 1450, 120ms
POST /api/analytics :done, 1580, 70ms section API Service /api/checkout/state :crit, 240,
440ms cache.get session#58:[redacted] :360, 10ms db.query select from session :370,
20ms db.query select from user :390, 20ms db.query select from checkout :410, 20ms
http.request GET http#58;//payments/poll :450, 210ms thread.spawn refresh-checkout-
cache :done, 670, 220ms section Payment Service /poll :crit, 470, 180ms db.query select
from payment :490, 30ms db.query update payment :530, 60ms
```

which has the following code:

```
gantt
  title Example Starfish Trace
  dateFormat x
  axisFormat %S.%L
```

```
section Frontend
/checkout :crit, 0, 1500ms
GET /api/session :150, 170ms
POST /api/analytics :190, 70ms
GET /api/checkout/state :200, 500ms
GET /api/checkout/cart :1100, 140ms
App :1300, 180ms
POST /api/analytics :done, 1450, 70ms
GET /assistant/poll :done, 1450, 120ms
POST /api/analytics :done, 1580, 70ms
```

```
section API Service
/api/checkout/state :crit, 240, 440ms
cache.get session#58;[redacted] :360, 10ms
db.query select from session :370, 20ms
db.query select from user :390, 20ms
db.query select from checkout :410, 20ms
http.request GET http#58;//payments/poll :450, 210ms
thread.spawn refresh-checkout-cache :done, 670, 220ms
```

```
section Payment Service
/poll :crit, 470, 180ms
db.query select from payment :490, 30ms
db.query update payment :530, 60ms
```

[Posts RSS](#)