

intro to tracing-based SLOs

Shelby Spees

Site Reliability Engineer

Equinix

@shelbyspees at #SLOconf

why tracing?

0.2525ms

0.3693ms

35.2μs

0.2415ms

36.3μs

34.5μs

34.1μs

use OpenTelemetry

auto-instrumentation for HTTP and gRPC

- request duration
- status codes
- client calls vs. server responses

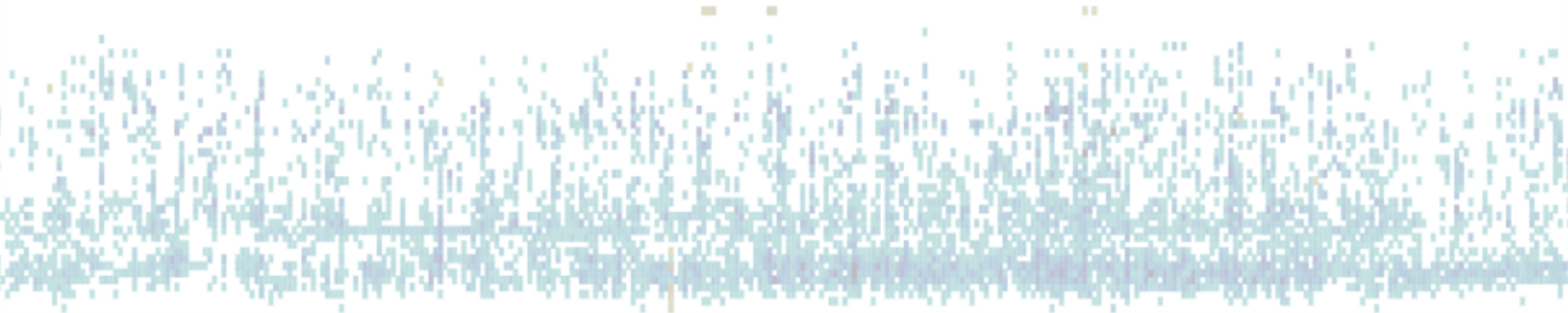
traces are fancy structured logs

```
{  
  "Timestamp": "2022-04-15T18:24:32.987575281Z",  
  "duration_ms": 42.076192,  
  "http.method": "GET",  
  "http.scheme": "https",  
  "http.status_code": 200,  
  "http.host": "api.awesome-service.net",  
  "http.target": "/all-the-things",  
  "service.name": "awesome-service",  
  "trace.trace_id": "11d8692d1cb9d55436c1a656999e0608",  
  "trace.span_id": "4f87aa2321768f18",  
  "trace.parent_id": "3a7caf3aaba06a5f"  
}
```

observability == exploration

- broad view: what's slow?
- zoom in: what's different about this slow traffic?
- zoom out: is this kind of traffic always slow?
- zoom in...

tracing-based SLOs



@shelbyspees at #SLOconf

instrument your code

```
func (m Action) BootDeviceSet(ctx context.Context, ...) (result string, err error) {
    tracer := otel.Tracer("pbnj")
    ctx, span := tracer.Start(ctx, "client.SetBootDevice", trace.WithAttributes(
        attribute.String("bmc.device", device),
        attribute.Bool("bmc.persistent", persistent),
        attribute.Bool("bmc.efiBoot", efiBoot),
    ))
    defer span.End()
    // . . . ° ☆ . . . ☆ . ° . .
    // ✧ set boot device ✧
    // . . . ° ☆ . . . ☆ . ° . .
}
```

defining a tracing-based SLI

- filter to what's relevant
- define a condition for "good"

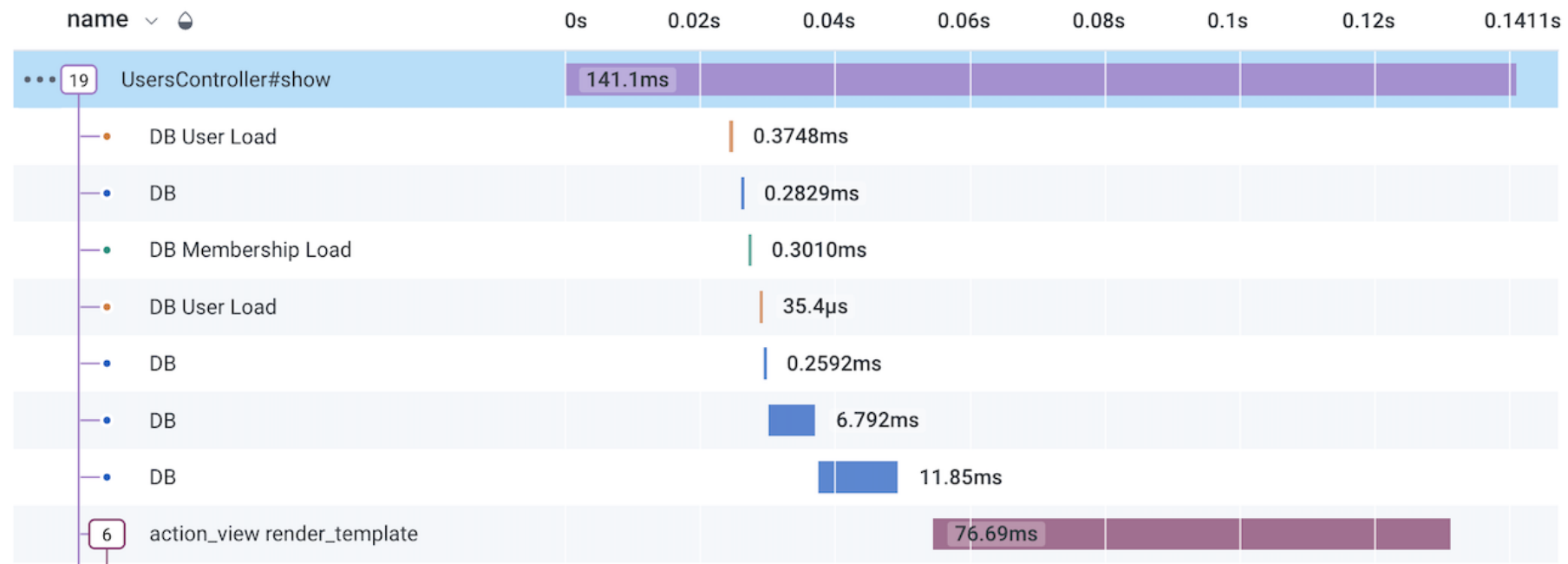
SLI: overall traffic



filter to what's relevant

```
IF(  
    // root spans  
    trace.parent_id == undefined  
    AND  
    // responding to client calls  
    span.kind == "server"  
)
```

root span of the trace



@shelbyspees at #SLOconf

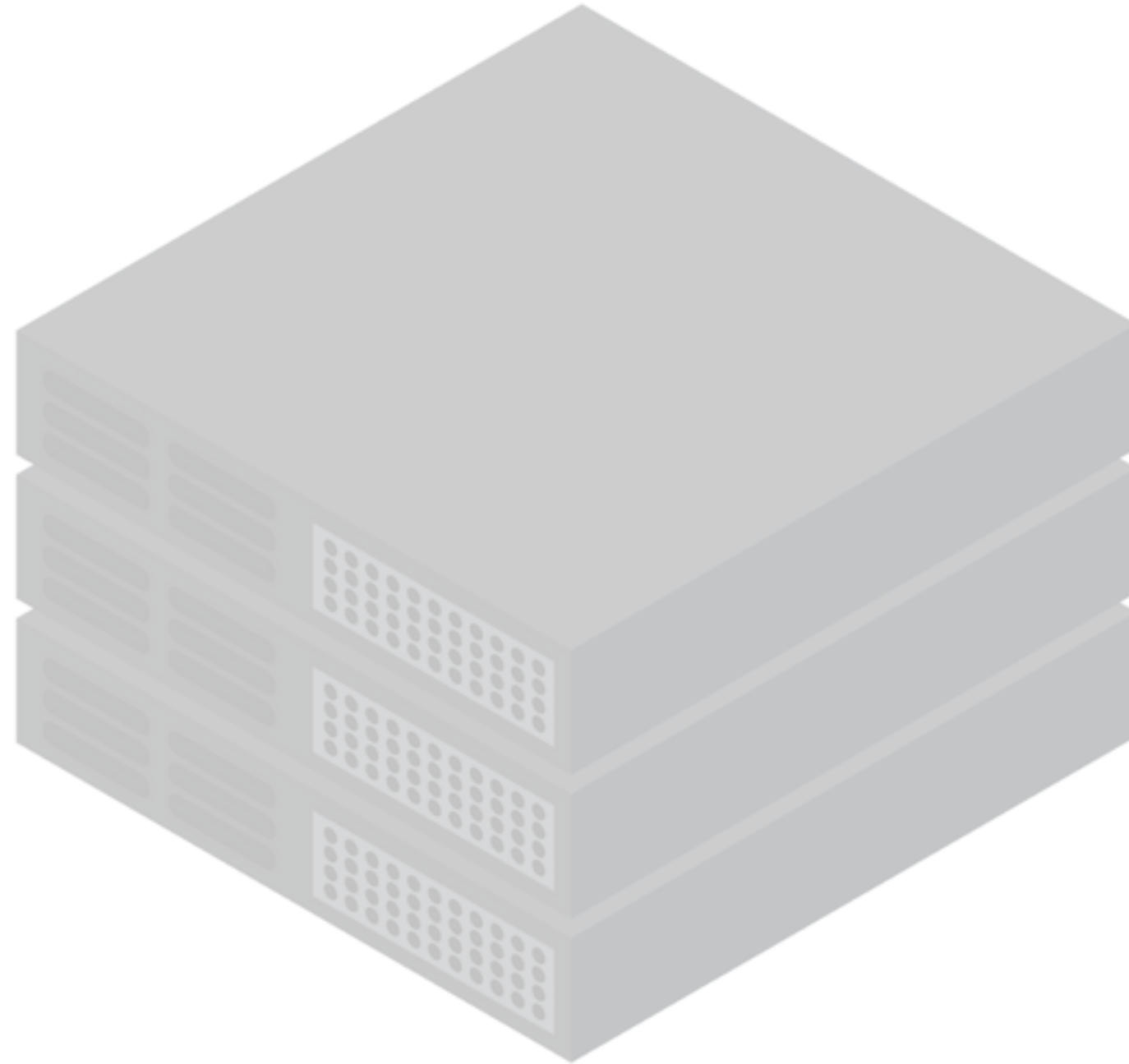
filter to what's relevant

```
IF(  
    trace.parent_id == undefined  
        AND  
    span.kind == "server"  
        AND  
    // filter out employee traffic  
    user.staff != true  
        AND  
    // filter out internal bot traffic  
    user.bot != true  
)
```

define a condition for “good”

```
IF(  
    // should not return a server error  
    http.status_code < 500  
    AND  
    // should return in less than 1s  
    duration_ms < 1000  
)
```

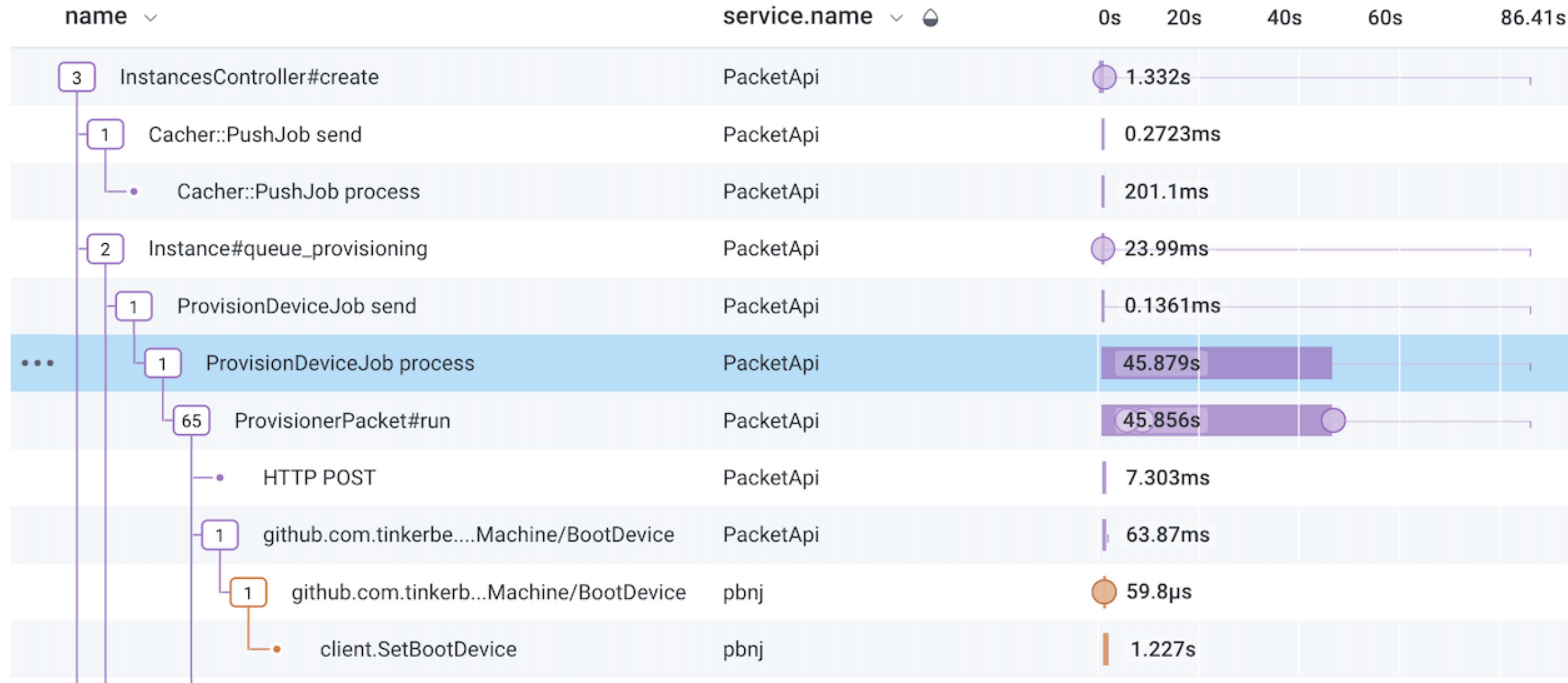
SLI: provisioning



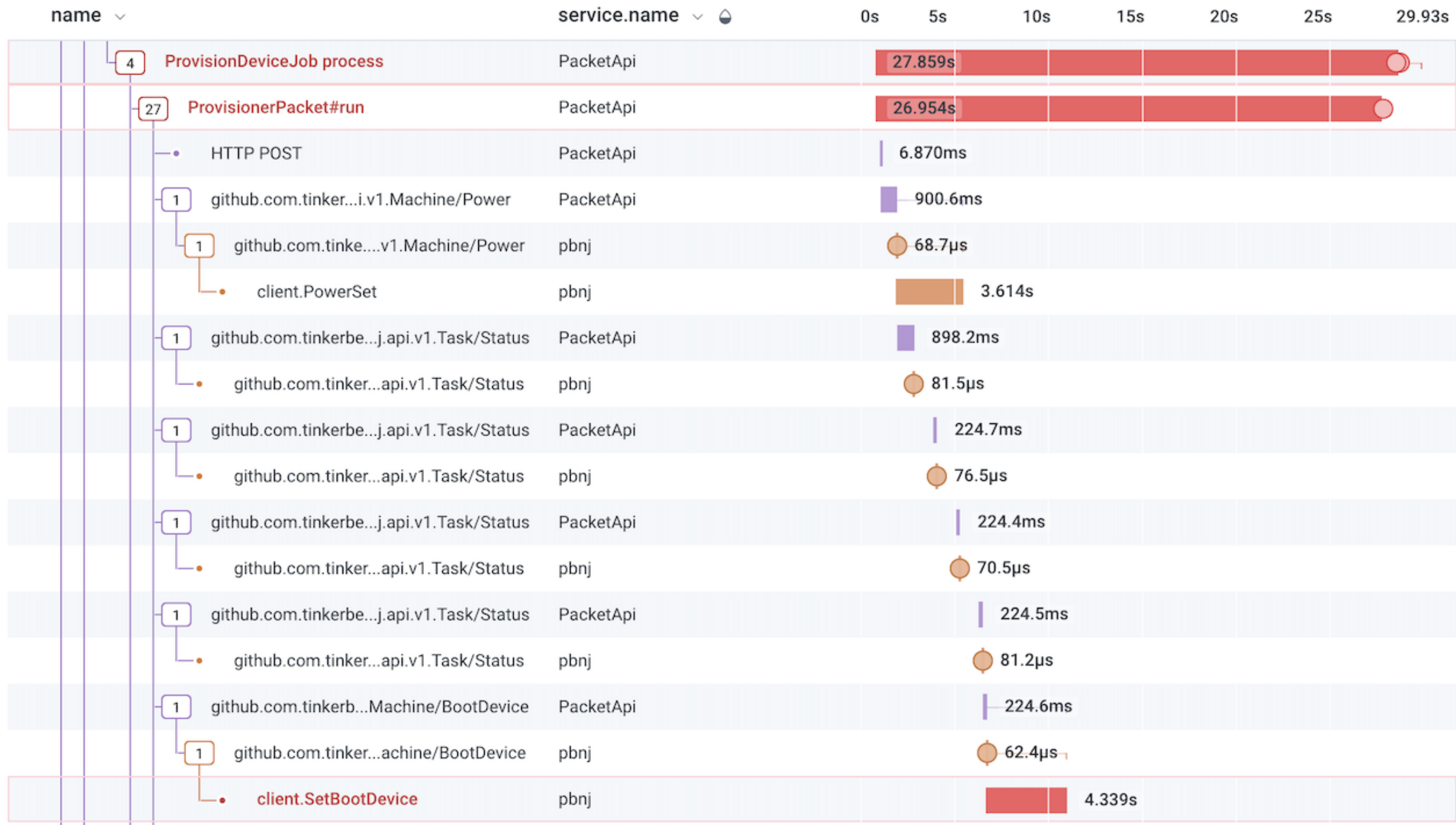
filter to what's relevant

```
IF(  
    // just the provision job  
    name == "ProvisionDeviceJob process"  
)
```

ProvisionDeviceJob



@shelbyspees at #SLOconf



@shelbyspees at #SLOconf

filter to what's relevant

```
IF(  
    // just the provision job  
    name == "ProvisionDeviceJob process"  
    AND  
    // filter out internal test projects  
    project.test != true  
)
```

define a condition for “good”

```
IF(  
    // should not fail  
    error != true  
)
```

you can do this!

- add OpenTelemetry auto-instrumentation
- observe and learn
- define basic SLIs
- add **custom** instrumentation
- observe and learn
- define ✨ fancy ✨ SLIs

thanks for watching!

@shelbyspees at #SLOconf