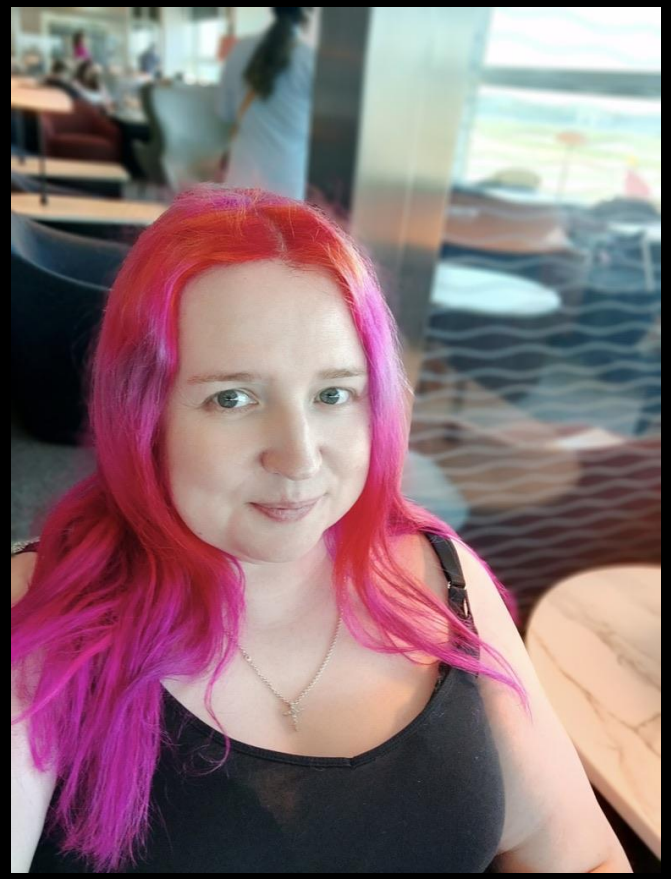
Observing your PHP application with OpenTelemetry

Jessica Smith, July 2025

@mintopia@phpc.social

About Me

- PHP user since 1998
- Originally from Gloucester, now in Reading
- Senior Principal Software Engineer for DigiCert
- Senior Developer at Fasthosts in Gloucester
- LAN Party Gamer



What we're going to cover

- What is observability
- Introducing OpenTelemetry
- How OpenTelemetry works
- Live Demo – Add OpenTelemetry to an Application
- Next Steps

What is Observability?

What is Observability?

Gaining understanding of a system by observing and measuring its outputs.

Why do we want this?

- Detecting Problems
- Quicker and easier troubleshooting
- Identifying Bottlenecks

Logs

- PSR-3
- Monolog
- Centralised – Logstash, Splunk

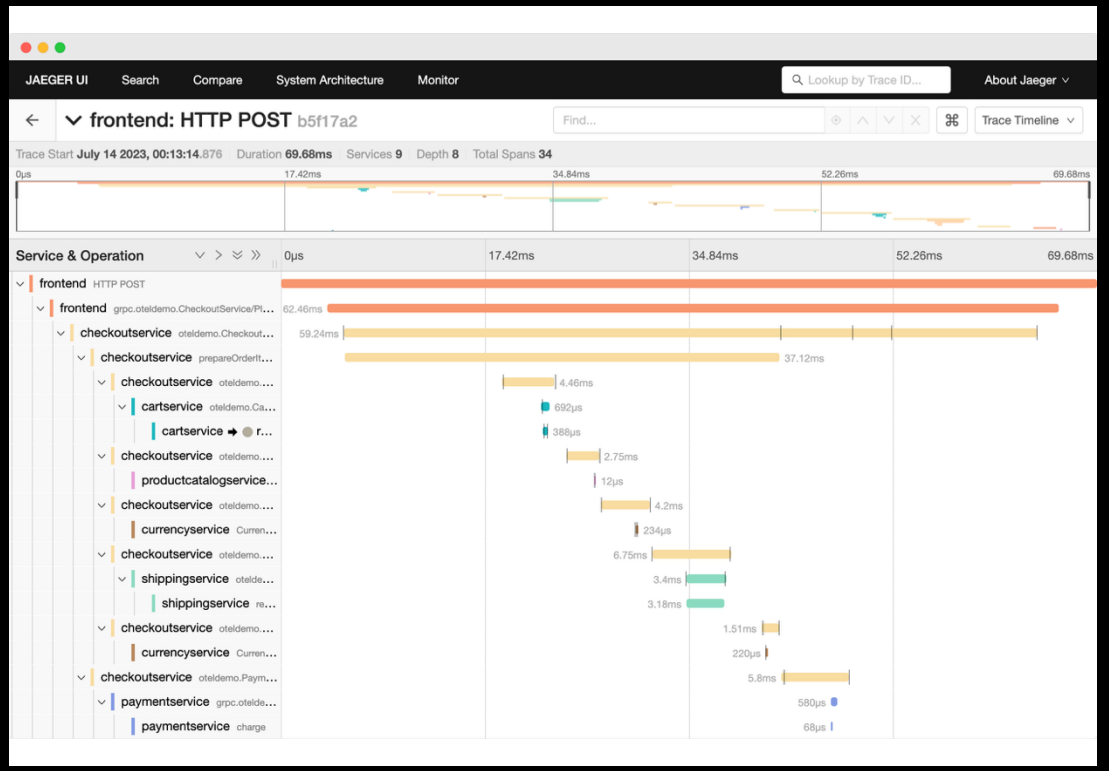
```
[2025-07-04 10:30:47] production.DEBUG: [Party:1] SLAN: Received simple webhook event
[2025-07-04 10:30:47] production.DEBUG: [Party:1] SLAN: Updating from PartyUpdate job
[2025-07-04 10:30:47] production.INFO: [Party:1] SLAN: Updating state
[2025-07-04 10:30:47] production.DEBUG: [Party:1] SLAN: Updating current song
[2025-07-04 10:30:47] production.DEBUG: [User:1] Mintopia: Spotify API -> getMyCurrentPlaybackInfo()
[2025-07-04 10:30:47] production.DEBUG: [User:1] Mintopia: Creating new API connection
[2025-07-04 10:30:47] production.INFO: [Party:1] SLAN: Updating current song to [Song:3374] Drop It Like It's Hot
[2025-07-04 10:30:47] production.DEBUG: [Party:1] SLAN: Spotify API -> getTrack(2NBQmPrOEEjA8VbeWOQGxO)
[2025-07-04 10:30:48] production.DEBUG: [Party:1] SLAN Updating history
[2025-07-04 10:30:48] production.DEBUG: [User:1] Mintopia: Spotify API -> getPlaylist(3pHPgF7cgkZfdn1uJbGGGm)
[2025-07-04 10:30:48] production.DEBUG: [Party:1] SLAN: Spotify API -> deletePlaylistTracks(3pHPgF7cgkZfdn1uJbGGGm, [])
[2025-07-04 10:30:48] production.DEBUG: [User:1] Mintopia: Spotify API -> getPlaylist(3pHPgF7cgkZfdn1uJbGGGm)
[2025-07-04 10:30:48] production.DEBUG: [Party:1] SLAN: Updating playlist
[2025-07-04 10:30:48] production.DEBUG: [Party:1] SLAN: Spotify API -> addPlaylistTracks(3pHPgF7cgkZfdn1uJbGGGm, [])
[2025-07-04 10:30:48] production.INFO: [Party:1] SLAN: Adding [Song:3442] 4 Chords to queue
[2025-07-04 10:30:48] production.DEBUG: [Party:1] SLAN: Spotify API -> queue(1cu5PHumWAqQf9M6BHQ8b7, [])
```

Metrics

- Key Performance Indicators
- Many different metrics:
  - Response Time
  - Exceptions
  - Successful/Failed Logins
  - So many things …
- Prometheus

Traces

- Following request from start to finish
- If using microservices – across microservices
- Measure performance of operations (spans) within a trace



Bringing it all together

- Three Pillars
    - Logs – The textual narrative
    - Metrics – Quantitative data
    - Traces – Visualising the flow of a request
- Combined, we get a holistic view with context

Observability Tools and Services

Self-Hosted

- Jaegar
- Uptrace
- OpenObserve
- SigNoz

Services

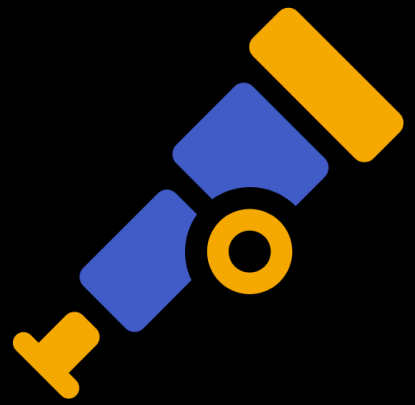
- Honeycomb
- Datadog
- New Relic
- Splunk
- AWS Cloud Watch
- So many…

How do we implement it?

- Platform and vendor agnostic
- Minimal effort
- Some sort of standard

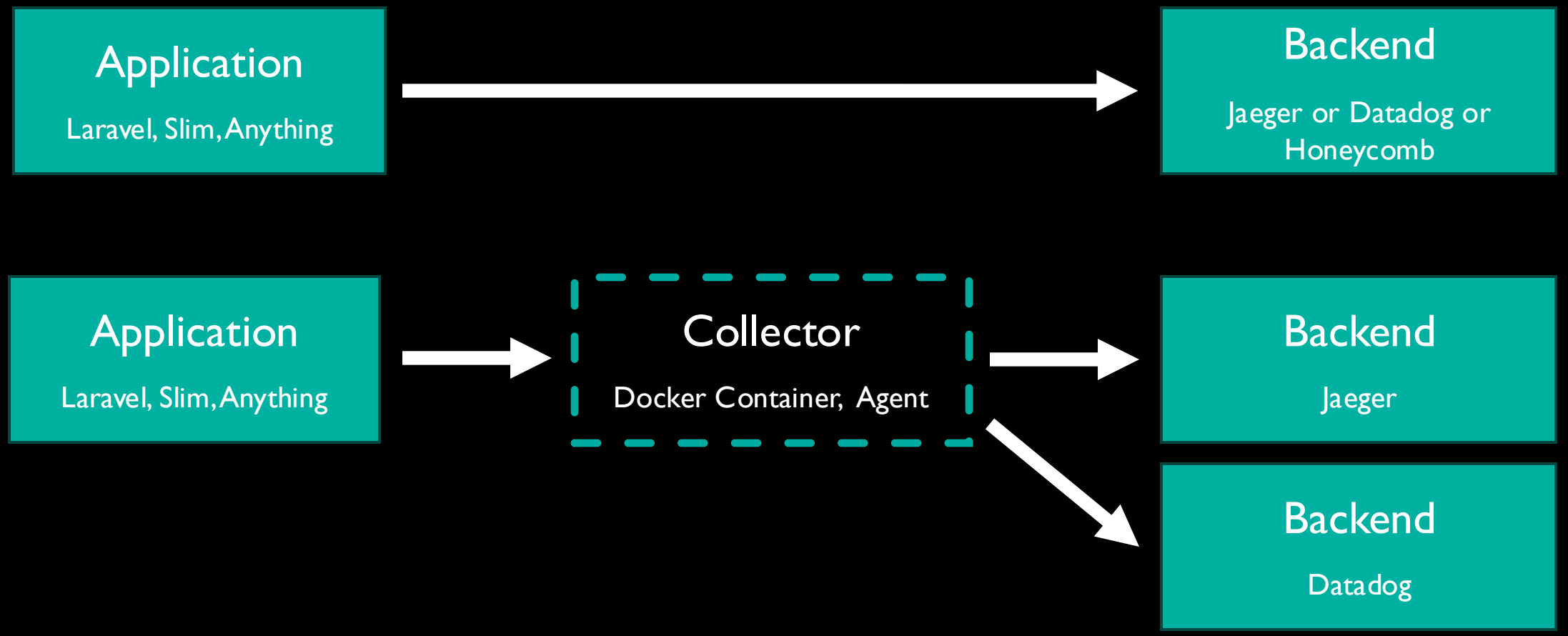
I have a suggestion…

OpenTelemetry



- Open Source
- CNCF Incubating Project
- Vendor Neutral
- Well Supported
- Zero-code required

How it works


Application
Laravel, Slim, Anything
Backend
Jaeger or Datadog or Honeycomb
Application
Laravel, Slim, Anything
Collector
Docker Container, Agent
Backend
Jaeger
Backend
Datadog

Collector

- Agent or Docker Container
- YAML Config
- Defines pipelines for data

- Extensions and Filters

Zero Code?

1. PHP Extension
2. Composer Packages
3. Environment Variables

PHP Extension

- Based on zend_observer
- Allows hooks on function and method calls
- Install through PECL, php-extension-installer or PIE

```
pie install open-telemetry/ext-opentelemetry
```

Composer Packages

- Uses the hooks from the extension
- Support for many different frameworks, PSRs, packages

```
composer require open-telemetry/sdk
composer require open-telemetry/opentelemetry-auto-Laravel
composer require open-telemetry/opentelemetry-auto-psr15
```

Environment Variables

Set using docker run or DotEnv

```
OTEL_PHP_AUTOLOAD_ENABLED=true
OTEL_SERVICE_NAME=application
OTEL_EXPORTER_OTLP_ENDPOINT=http://collector:4318
OTEL_EXPORTER_OTLP_PROTOCOL=http/json
```

Demo

Let's do this live!

Going further

- Manual Instrumentation
- Use attribute based hooks
- Write your own hooks for your application
- Exporting metrics
  - Custom integration?
  - Prometheus?

Attribute Based Hooks

- Enable them in php.ini

```php
<?php

use OpenTelemetry\API\Instrumentation\WithSpan;

class MyClass
{
    #[WithSpan]
    public function expensiveFunction(): void
    {
        doSomethingThatIWantToMeasure();
    }
}
```

Sampling

- Probably don't want all the data
- 5% of all data is OK
- Include all errors

```
OTEL_TRACES_SAMPLER=traceidratio
OTEL_TRACES_SAMPLER_ARG=0.2
```

- More complex sampling in the collector

Resources

- OpenTelemetry - https://opentelemetry.io
- PHP Installer for Extensions - https://github.com/php/pie
- Slides - https://github.com/mintopia/php-opentelemetry-talk-2025

Thank you

https://phpc.social/@mintopia

https://github.com/mintopia

https://discord.gg/roave

Slides