Northwestern Mutual®

RESTifying OpenTelemetry

Justin Scherer
Justin Snyder

Who Are We?

Justin Scherer
Northwestern Mutual Life
Lead Software Engineer

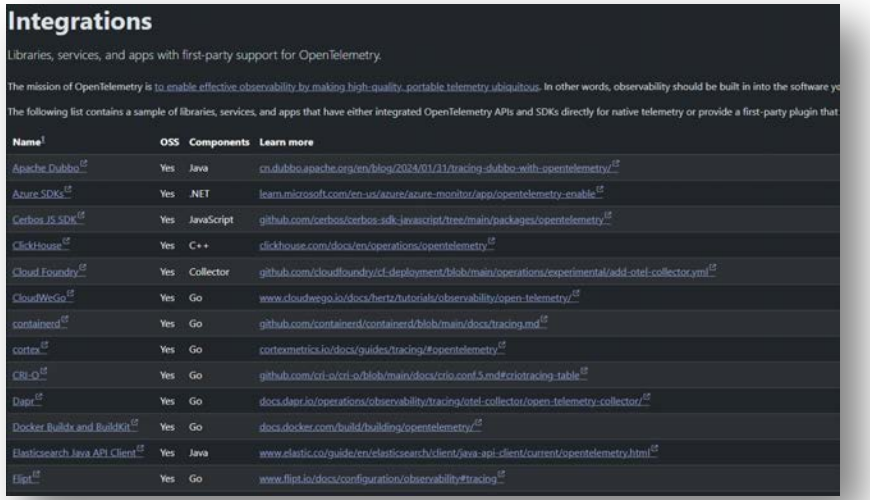
Justin Snyder
Northwestern Mutual Life
Manager Software Engineering

OpenTelemetry

Credit to OpenTelemetry Team

- Easily create traces and metrics to ingest into your APM of choice
- Easily ingest logs into your APM
- OpenSource and extensible
- Supports many popular languages

Language	Traces	Metrics	Logs
C++	Stable	Stable	Stable
C#/.NET	Stable	Stable	Stable
Erlang/Elixir	Stable	Experimental	Experimental
Go	Stable	Stable	Alpha
Java	Stable	Stable	Stable
JavaScript	Stable	Stable	Experimental
PHP	Stable	Stable	Stable
Python	Stable	Stable	Experimental
Ruby	Stable	In development	In development
Rust	Beta	Alpha	Alpha
Swift	Stable	Experimental	In development



Integrations

Libraries, services, and apps with first-party support for OpenTelemetry.

The mission of OpenTelemetry is to enable effective observability by making high-quality, portable telemetry ubiquitous. In other words, observability should be built in into the software y

The following list contains a sample of libraries, services, and apps that have either integrated OpenTelemetry APIs and SDKs directly for native telemetry or provide a first-party plugin that

Name	OSS	Components	Learn more
Apache Dubbo	Yes	Java	cn.dubbo.apache.org/en/blog/2024/01/31/tracing-dubbo-with-opentelemetry/
Azure SDKs	Yes	.NET	learn.microsoft.com/en-us/azure/azure-monitor/app/opentelemetry-enable
Cerbos JS SDK	Yes	JavaScript	github.com/cerbos/cerbos-sdk-javascript/tree/main/packages/opentelemetry
ClickHouse	Yes	C++	clickhouse.com/docs/en/operations/opentelemetry
Cloud Foundry	Yes	Collector	github.com/cloudfoundry/cf-deployment/blob/main/operations/experimental/add-otel-collector.yml
CloudWeGo	Yes	Go	www.cloudwego.io/docs/hertz/tutorials/observability/open-telemetry/
containerd	Yes	Go	github.com/containerd/containerd/blob/main/docs/tracing.md
cortex	Yes	Go	cortexmetrics.io/docs/guides/tracing/#opentelemetry
CRI-O	Yes	Go	github.com/cri-o/cri-o/blob/main/docs/crio.conf.5.md#criotracing-table
Dapr	Yes	Go	docs.dapr.io/operations/observability/tracing/otel-collector/open-telemetry-collector/
Docker Buildx and BuildKit	Yes	Go	docs.docker.com/build/building/opentelemetry/
Elasticsearch Java API Client	Yes	Java	www.elastic.co/guide/en/elasticsearch/client/java-api-client/current/opentelemetry.html
Flipt	Yes	Go	www.flipt.io/docs/configuration/observability#tracing

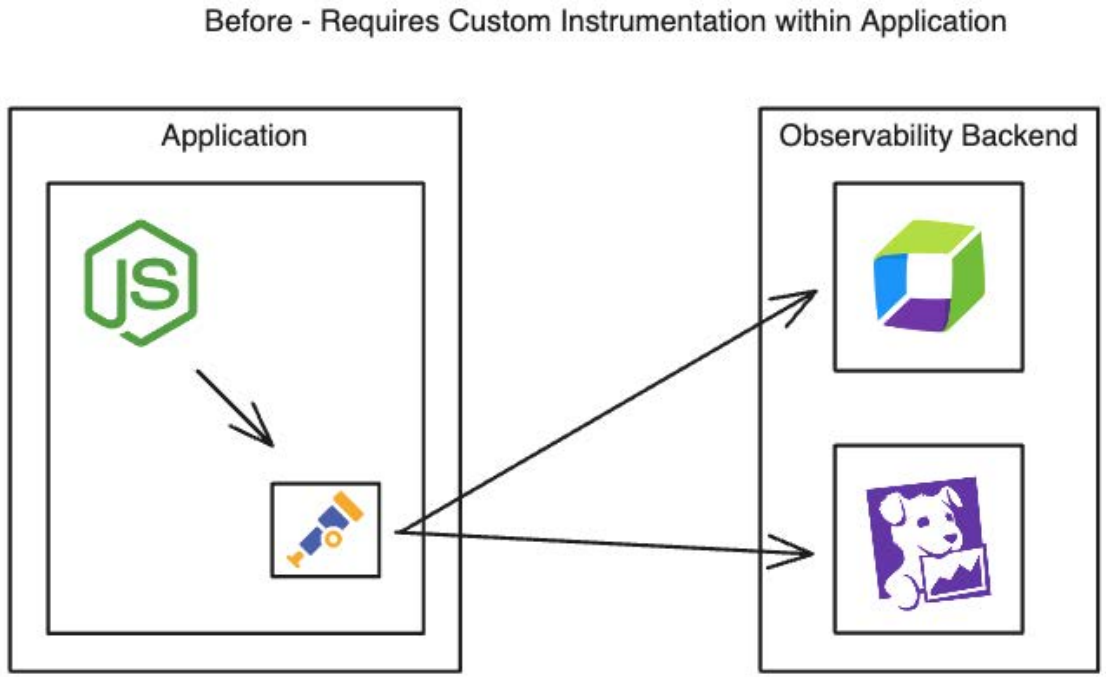
Issues Still Facing OpenTelemetry

- Working with it is easy for developers, but not for everyone else
- Integration into legacy technologies is sparse (check out OpenMainframe if interested in this!)
- Manual/human processes are hard to capture with OpenTelemetry

What if we could provide a simple to use system to develop traces for the above?

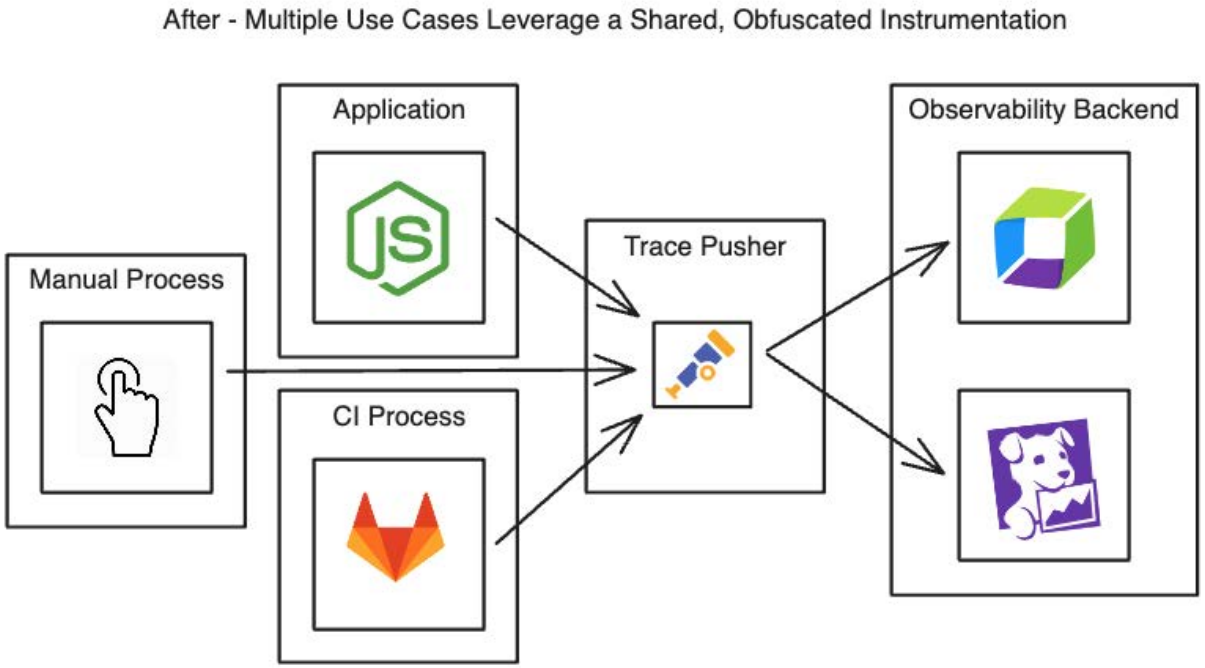
OpenTelemetry TracePusher API

- Solution - Obfuscate the instrumentation behind an open interface (API / Module)
- Avoids complicated HTTP Request Formatting with Protocol Buffers and Ingestion API Responses
- Remove need to maintain copies OpenTelemetry Schemas within your Application



OpenTelemetry TracePusher API

- Solution - Obfuscate the instrumentation behind an open interface (API / Module)
- Avoids complicated HTTP Request Formatting with Protocol Buffers and Ingestion API Responses
- Remove need to maintain copies OpenTelemetry Schemas within your Application



Introducing OTEL rest-trace Service!

Use cases for OTEL rest-trace Service

- Hook into technology that has no SDK built out for it (simple REST calls)
- Manual processes can be tracked (delivery services, phone calls, etc.)
- No-code solutions through REST calls
- CI/CD pipelines

Code-Level Demos – Node.JS

Generic API Demo

```
import { Router } from 'express';
const router = Router();

/* Business Logic API */
router.get('/', (req, res) => {

  // Call TracePusher to Start Trace - {api}/trace/start
  const { traceId } = fetch('https://internal.domain.com/trace-pusher/trace/start', {
    method: 'POST',
    body: { properties: { 'service.name': 'Business Logic API' }},
  });

  // Start Your Business Logic, ensuring `traceId` value is accessible
  businessLogicA(traceId);
  businessLogicB(traceId);

  // Call TracePusher to End Trace and Push to Observability Backend - {api}/trace/{traceId}/end
  fetch(`https://internal.domain.com/trace-pusher/trace/${traceId}/end`, {
    method: 'POST',
  });
});

export default router;
```

Generic API Demo

```
const businessLogicA = (traceId) => {
  // Call TracePusher to Start Span - {api}/span/{traceId}/start
  const { spanId } = fetch(`https://internal.domain.com/trace-pusher/span/${traceId}/start`, {
      method: 'POST',
      body: {
        name: 'BusinessLogicPartOne',
        properties: { start: 'start' },
      },
    }
  );

  // Do First Section of Business Logic
  console.log('Do Business Part One.');

  // Call TracePusher to End Span - {api}/span/{traceId}/end/{spanId}
  fetch(`https://internal.domain.com/trace-pusher/span/${traceId}/end/${spanId}`, {
    method: 'POST',
    body: {
      msg: 'BusinessLogicPartOne Complete',
    },
  });
};
```

Generic API Demo

```
import { Router } from 'express';
const router = Router();

/* Business Logic API */
router.get('/', (req, res) => {

  // Call TracePusher to Start Trace - {api}/trace/start
  const { traceId } = fetch('https://internal.domain.com/trace-pusher/trace/start', {
    method: 'POST',
    body: { properties: { 'service.name': 'Business Logic API' }},
  });

  // Start Your Business Logic, ensuring `traceId` value is accessible
  businessLogicA(traceId);
  businessLogicB(traceId);

  // Call TracePusher to End Trace and Push to Observability Backend - {api}/trace/{traceId}/end
  fetch(`https://internal.domain.com/trace-pusher/trace/${traceId}/end`, {
    method: 'POST',
  });
});

export default router;
```

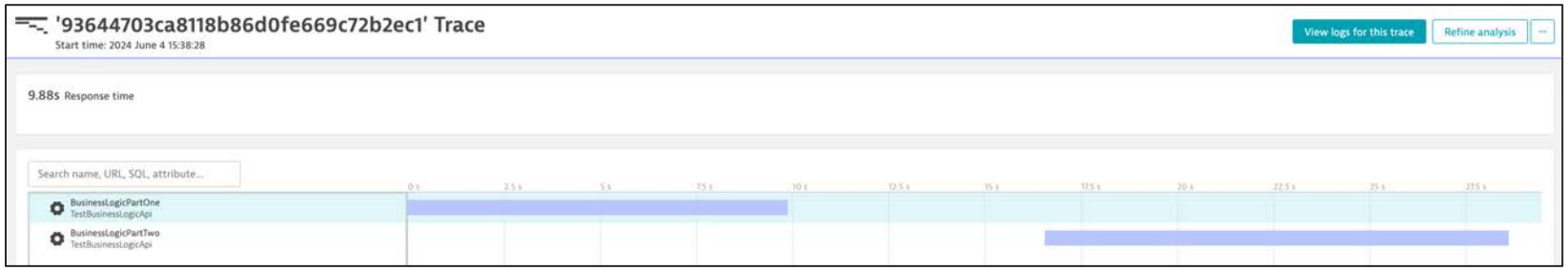
Generic API Demo

```
const businessLogicB = (traceId) => {
  // Call TracePusher to Start Span - {api}/span/{traceId}/start
  const { spanId } = fetch(`https://internal.domain.com/trace-pusher/span/${traceId}/start`, {
      method: 'POST',
      body: {
        name: 'BusinessLogicPartTwo',
        properties: { start: 'start' },
      },
    }
  );

  // Do Second Section of Business Logic
  console.log('Do Business Part Two.');

  // Call TracePusher to End Span - {api}/span/{traceId}/end/{spanId}
  fetch(`https://internal.domain.com/trace-pusher/span/${traceId}/end/${spanId}`, {
    method: 'POST',
    body: {
      msg: 'BusinessLogicPartTwo Complete',
    },
  });
};
```

Generic API Demo

```
import { Router } from 'express';
const router = Router();

/* Business Logic API */
router.get('/', (req, res) => {

  // Call TracePusher to Start Trace - {api}/trace/start
  const { traceId } = fetch('https://internal.domain.com/trace-pusher/trace/start', {
    method: 'POST',
    body: { properties: { 'service.name': 'Business Logic API' }},
  });

  // Start Your Business Logic, ensuring `traceId` value is accessible
  businessLogicA(traceId);
  businessLogicB(traceId);

  // Call TracePusher to End Trace and Push to Observability Backend - {api}/trace/{traceId}/end
  fetch(`https://internal.domain.com/trace-pusher/trace/${traceId}/end`, {
    method: 'POST',
  });
});

export default router;
```

Generic API Demo



BusinessLogicPartOne

Summary | Timing | Events (0)

Topology

Service	TestBusinessLogicApi

Metadata

Endpoint	BusinessLogicPartOne
Trace ID	93644703ca8118b86d0fe669c72b2ec1
Span ID	8fd58c4b22a610d4
Span kind	Client
Instrumentation scope	otel.rest.tracepusher
Status message	BusinessLogicPartOne Complete

Attributes

start	start

Resource attributes

service.name	TestBusinessLogicApi

Dynatrace internal-use attributes

dt.entity.service	SERVICE-B70AC43E3C4AD228
dt.internal.failure_detection_verdict	SUCCESS
endpoint.name	BusinessLogicPartOne
request.is_failed	false
request.is_root_span	true
supportability.endpoint_name_rule	span.name fallback

BusinessLogicPartTwo

Summary | Timing | Events (0)

Topology

Service	TestBusinessLogicApi

Metadata

Endpoint	BusinessLogicPartTwo
Trace ID	93644703ca8118b86d0fe669c72b2ec1
Span ID	95ef8471ce2c6ef2
Span kind	Client
Instrumentation scope	otel.rest.tracepusher
Status message	BusinessLogicPartTwo Complete

Attributes

start	start

Resource attributes

service.name	TestBusinessLogicApi

Dynatrace internal-use attributes

dt.entity.service	SERVICE-B70AC43E3C4AD228
dt.internal.failure_detection_verdict	SUCCESS
endpoint.name	BusinessLogicPartTwo
request.is_failed	false
request.is_root_span	true
supportability.endpoint_name_rule	span.name fallback

Code-Level Demos – Gitlab CI

CI Process Demo

```
stages:
  - .pre
  - Build
  - Test
  - .post
```

```
Start Trace:
  stage: .pre
  script:
  # Call TracePusher to Start Trace - {api}/trace/start
    - >
      curl -X POST https://internal.domain.com/trace-pusher/trace/start \
        -D '{ "properties": { "service.name": "CI Process" } }' \
        > $CI_PROJECT_DIR/trace.info
  # Ensure `traceId` value is accessible
  artifacts:
    paths:
      - trace.info
```

CI Process Demo

```yaml
stages:
  - .pre
  - Build
  - Test
  - .post
```

```yaml
Build:
  stage: Build
  before_script:
    # Call TracePusher to Start Span - {api}/trace-pusher/span/{traceId}/start
    - TRACE_ID=$(cat trace.info | jq -r '.traceId')
    - >
      SPAN_ID=$( curl -X POST https://internal.domain.com/trace-pusher/span/$TRACE_ID/start \
        -D '{ "name": "Build", "properties": { "start": "start", "jobId": 12345 } }' \
        | jq -r '.spanId'
      )
  script:
    - echo "Building the project"
  after_script:
    # Call TracePusher to End Span - {api}/trace-pusher/span/{traceId}/{spanId}/end
    - >
      curl -X POST https://internal.domain.com/trace-pusher/span/$TRACE_ID/end/$SPAN_ID \
        -D '{ "msg": "Build Complete" }'
```

CI Process Demo

```yaml
stages:
  - .pre
  - Build
  - Test
  - .post
```

```yaml
Test:
  stage: Test
  before_script:
    # Call TracePusher to Start Span - {api}/trace-pusher/span/{traceId}/start
    - TRACE_ID=$(cat trace.info | jq -r '.traceId')
    - >
      SPAN_ID=$( curl -X POST https://internal.domain.com/trace-pusher/span/$TRACE_ID/start \
        -D '{ "name": "Test", "properties": { "start": "start", "jobId": 67890 } }' \
        | jq -r '.spanId'
      )
  script:
    - echo "Testing the project"
  after_script:
    # Call TracePusher to End Span - {api}/trace-pusher/span/{traceId}/{spanId}/end
    - >
      curl -X POST https://internal.domain.com/trace-pusher/span/$TRACE_ID/end/$SPAN_ID \
        -D '{ "msg": "Test Complete" }'
```

CI Process Demo

```
stages:
  - .pre
  - Build
  - Test
  - .post
```

```
Post Trace:
  stage: .post
  script:
  # Call TracePusher to End Trace - {api}/trace/end
    - >
      curl -X POST https://internal.domain.com/trace-pusher/trace/$TRACE_ID/end
```

CI Process Demo

Build

Summary | Timing | Events (0)

Topology

Service	CI Process

Metadata

Endpoint	Build
Trace ID	7e2dc21f3b0c74c6c8c8cb1b0426f9ef
Span ID	0aebcb8cbe30ed61
Span kind	Client
Instrumentation scope	otel.rest.tracepusher
Status message	Build Complete

Attributes

jobId	12345
start	start

Resource attributes

service.name	CI Process

Dynatrace internal-use attributes

dt.entity.service	SERVICE-C9E0061E35F6F07D
dt.internal.failure_detection_verdict	SUCCESS
endpoint.name	Build
request.is_failed	false
request.is_root_span	true
supportability.endpoint_name_rule	span.name fallback

Test

Summary | Timing | Events (0)

Topology

Service	CI Process

Metadata

Endpoint	Test
Trace ID	7e2dc21f3b0c74c6c8c8cb1b0426f9ef
Span ID	796c8d9b4b0be918
Span kind	Client
Instrumentation scope	otel.rest.tracepusher
Status message	Test Complete

Attributes

jobId	67890
start	start

Resource attributes

service.name	CI Process

Dynatrace internal-use attributes

dt.entity.service	SERVICE-C9E0061E35F6F07D
dt.internal.failure_detection_verdict	SUCCESS
endpoint.name	Test
request.is_failed	false
request.is_root_span	true
supportability.endpoint_name_rule	span.name fallback

What are the next steps?

Next Steps

- Standardization around trace creation
- Easy of use (right now, 1x1 mapping of Otel keys to what is required in REST bodies)
- OpenSource track

Conclusion