Anatomy of a Python OpenTelemetry instrumentation

Riccardo Magliocchetti

FOSDEM, 2025-02-02

Riccardo Magliocchetti

Senior Software Engineer

Elastic Observability

Maintainer OpenTelemetry Python

elastic

Agenda

How OpenTelemetry Python works:

- opentelemetry-bootstrap to install instrumentation libraries

Agenda

How OpenTelemetry Python works:

- opentelemetry-bootstrap
- opentelemetry-instrument for automatic instrumentation

Agenda

How OpenTelemetry Python works:

- opentelemetry-bootstrap
- opentelemetry-instrument
- Entry points to discover components

Agenda

How OpenTelemetry Python works:

- opentelemetry-bootstrap
- opentelemetry-instrument
- Entry points
- Instrumentation libraries for third party code

Observability and OpenTelemetry

elastic

Observability

Observability is the ability to understand the internal state of a system by examining its outputs.

elastic

OpenTelemetry

An open source observability framework providing specifications and implementations in order to create and manage telemetry data:

- Traces: execution paths

OpenTelemetry

An observability framework providing specifications and implementations in order to create and manage telemetry data:

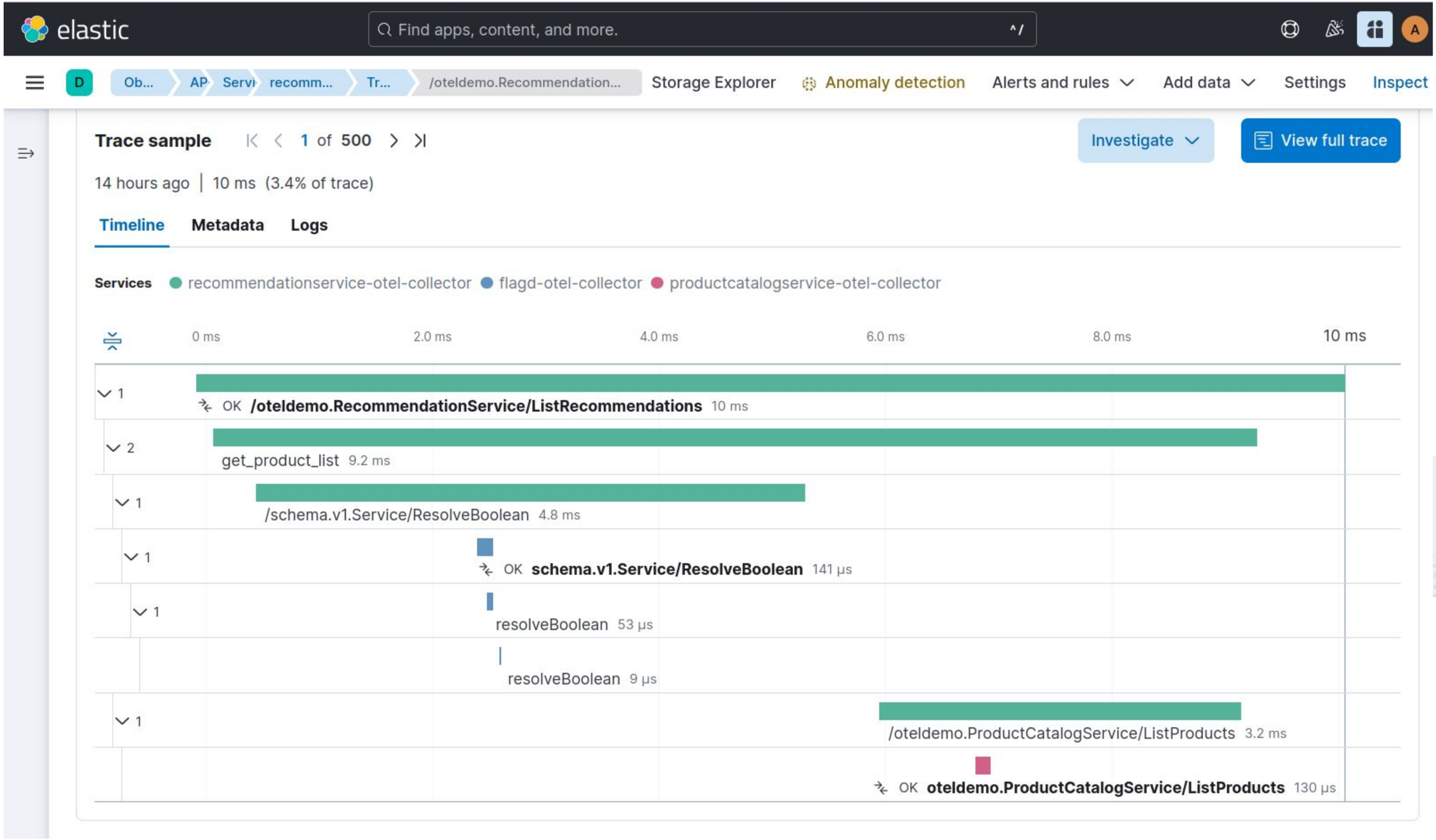
- Traces: execution paths
- Metrics: measurements

elastic

OpenTelemetry

An observability framework providing specifications and implementations in order to create and manage telemetry data:

- Traces: execution paths
- Metrics: measurements
- Logs: time stamped text


elastic
Find apps, content, and more.
^/
A
D
Ob...
AP
Servi
recomm...
Tr...
/oteldemo.Recommendation...
Storage Explorer
Anomaly detection
Alerts and rules
Add data
Settings
Inspect
Trace sample
1 of 500
Investigate
View full trace
14 hours ago | 10 ms (3.4% of trace)
Timeline
Metadata
Logs
Services
recommendationservice-otel-collector
flagd-otel-collector
productcatalogservice-otel-collector
0 ms
2.0 ms
4.0 ms
6.0 ms
8.0 ms
10 ms
1
OK /oteldemo.RecommendationService/ListRecommendations 10 ms
2
get_product_list 9.2 ms
1
/schema.v1.Service/ResolveBoolean 4.8 ms
1
OK schema.v1.Service/ResolveBoolean 141 µs
1
resolveBoolean 53 µs
resolveBoolean 9 µs
1
/oteldemo.ProductCatalogService/ListProducts 3.2 ms
OK oteldemo.ProductCatalogService/ListProducts 130 µs

elastic

OpenTelemetry Python

- opentelemetry-python
  - API: defines the interface

OpenTelemetry Python

- opentelemetry-python
  - API: defines the interface
  - SDK: an implementation of the API

elastic

OpenTelemetry Python

- opentelemetry-python
  - API: defines the interface
  - SDK: an implementation of the API
  - Semantic conventions: attributes definitions for traces, metrics, logs

OpenTelemetry Python

- opentelemetry-python
- opentelemetry-python-contrib
  - instrumentation*/opentelemetry-instrumentation-*: instrumentations libraries

OpenTelemetry Python

- opentelemetry-python
- opentelemetry-python-contrib
  - instrumentation*/opentelemetry-instrumentation-*
  - opentelemetry-instrumentation: auto-instrumentation

OpenTelemetry Python

- opentelemetry-python
- opentelemetry-python-contrib
  - instrumentation/opentelemetry-instrumentation-*
  - opentelemetry-instrumentation
  - opentelemetry-distro: default configurations

Demo application

elastic

Flask application

```
# app.py
from flask import Flask


app = Flask(__name__)


@app.route("/")
def hello_world():
    return "Hello, World!"
```

Flask application setup

```
$ python3 -m venv
$ . ./venv/bin/activate
$ pip install flask
# the distro depends on api, sdk and instrumentation packages
$ pip install opentelemetry-distro
```

opentelemetry-bootstrap

elastic

opentelemetry-bootstrap

CLI tool from opentelemetry-instrumentation package to list or install any instrumentation library available for your project dependencies.

opentelemetry-bootstrap

```
# opentelemetry.instrumentation.boostrap_gen
libraries = [
  {
    "library": "flask >= 1.0",
    "instrumentation": "opentelemetry-instrumentation-flask==0.50b0",
    }, ...
]
default_instrumentations = [
    "opentelemetry-instrumentation-asyncio==0.50b0",
    ...
]
```


elastic

opentelemetry-bootstrap

$ opentelemetry-bootstrap –action=install

this is the output of listing, install too verbose

opentelemetry-instrumentation-asyncio==0.50b0

...

opentelemetry-instrumentation-wsgi==0.50b0

opentelemetry-instrumentation-flask==0.50b0

...

elastic

opentelemetry-instrument

Auto instrumentation

elastic

opentelemetry-instrument

The tool used to implement auto instrumentation: the ability to setup OpenTelemetry tracing, logging and metering for your application without touching its code

sitecustomize?

elastic

sitecustomize

By default the site module of your python installation tries to load from your python path a module named sitecustomize.

Reference
https://docs.python.org/3/library/site.html#module-sitecustomize

sitecustomize.py

```
$ python3 -c 'print("world")'
world
```


elastic

sitecustomize.py

```
$ python3 -c 'print("world")'
world
$ PYTHONPATH=. python3 -c 'print("world")'
hello world
```


elastic

sitecustomize.py

```
$ PYTHONPATH=. python3 -c 'print("world")'
hello world
$ cat sitecustomize.py
print("hello", end=” “)
```


elastic

opentelemetry-instrument

- Loads the distro
- Loads the configuration
- Loads the instrumentations

opentelemetry-instrument

```
$ opentelemetry-instrument --traces_exporter=console --metrics_exporter=none --logs_exporter=none flask run
{
    "name": "GET /",
    "context": {
        "trace_id": "0x37b87d1a4361879180cd87eb082cf57b",
        "span_id": "0xf461041d79e0b85a",
        "trace_state": "[]"
    },
    "kind": "SpanKind.SERVER",
    "parent_id": null,
    "start_time": "2024-08-24T18:06:58.270986Z",
    "end_time": "2024-08-24T18:06:58.272434Z",
    ...
```

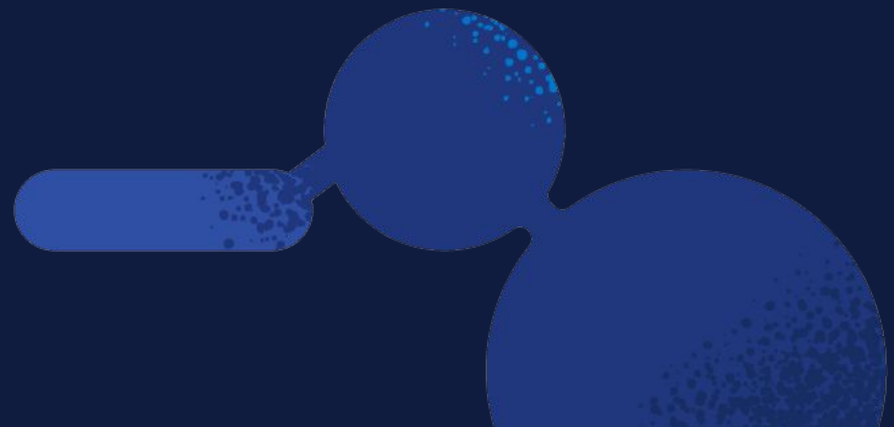

elastic

opentelemetry-instrument

```
$ opentelemetry-instrument --traces_exporter=console --metrics_exporter=none flask run
    ...
    "attributes": {
        "http.method": "GET",
        "http.server_name": "127.0.0.1",
        "http.scheme": "http",
        "net.host.name": "127.0.0.1:5000",
        ...
    }
}
```


elastic

Entry points



elastic

Entry points

Entry points are a mechanism that packages use to provide discoverability over some of their code.

Entry points

opentelemetry-distro/pyproject.toml

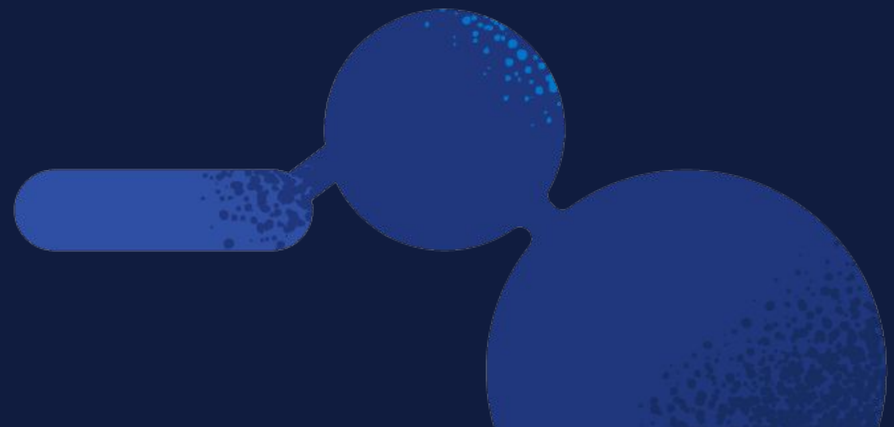
```
[project.entry-points.opentelemetry_configurator] # group
# name = code reference
configurator = "opentelemetry.distro:OpenTelemetryConfigurator"
```

Entry points

```
from opentelemetry.util._importlib_metadata import entry_points

for entry_point in entry_points(group="opentelemetry_configurator"):
    # call configure method of OpenTelemetryConfigurator instance
    entry_point.load()().configure()
    print(f"Configured: {entry_point.name}")
```

Patching third party code



elastic

Monkey patching

Monkey patching is a technique used to dynamically change the behavior of some code at runtime.

wrapt

Just use wrapt!

Documentation https://wrapt.readthedocs.io/en/master/

elastic

Patching example

From https://github.com/xrmx/opentelemetry-instrumentation-asgiref/

```
from wrapt import wrap_function_wrapper


class AsgirefInstrumentor(BaseInstrumentor):
    def _instrument(self, **kwargs):
        wrap_function_wrapper(asgiref.sync, "async_to_sync", self.__wrapper)
    def _uninstrument(self, **kwargs):
        unwrap(asgiref.sync, "async_to_sync")
```

Patching example

From https://github.com/xrmx/opentelemetry-instrumentation-asgiref/

```
class AsgirefInstrumentor(BaseInstrumentor):
    ...
    def __wrapper(self, wrapped, instance, args, kwargs):
        with self.tracer.start_as_current_span( “async_to_sync", kind=SpanKind.INTERNAL) as span:
            attributes = {
                "exception.type": “async_to_sync”,
                "exception.stacktrace": "\n".join(traceback.format_stack(limit=10))
            }
            span.add_event(name="exception", attributes=attributes)
        return wrapped(*args, **kwargs)
```

Patching example #2

From [opentelemetry-instrumentation-botocore]:

```
from wrapt import ObjectProxy

class ConverseStreamWrapper(ObjectProxy):
    # USAGE: result["stream"] = ConverseStreamWrapper(result["stream"])
    def __init__(self, stream: botocore.eventstream.EventStream):
        super().__init__(stream)


    def __iter__(self):
        for event in self.__wrapped__: # see first param of __init__
            self._process_event(event) # this is calling our own code
            yield event
```


elastic

Conclusions

elastic

Conclusions

- opentelemetry-bootstrap for installing instrumentations
- site and sitecustomize to run our code first
- Entry points for components discovery
- wrapt for third party code patching

Contacts

- https://opentelemetry.io
- opentelemetry-python and opentelemetry-python-contrib
- #otel-python on CNCF slack
- @rmistaken@hachyderm.io

Thank you!