KubeCon | CloudNativeCon

Europe 2020

Virtual

OpenTelemetry Auto-Instrumentation Deep Dive

Carlos Alberto Cortez & Alex Boten, Lightstep

What's OpenTelemetry?

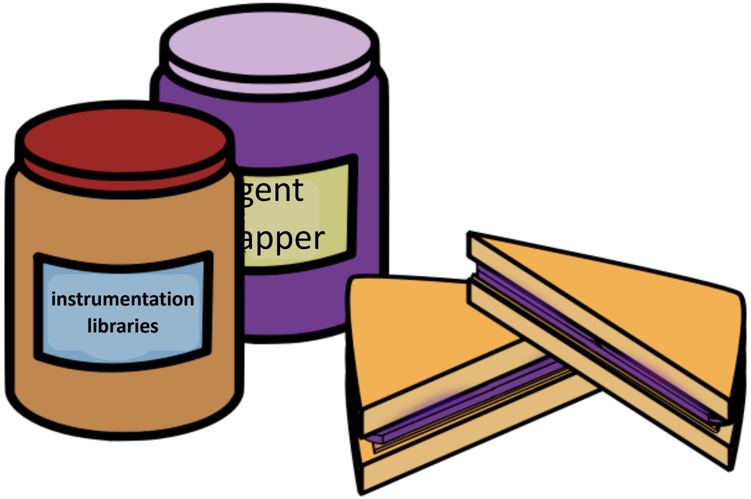
- Capture telemetry such as **distributed traces** and **metrics**
- Sends this data to backends
- Complete ecosystem for many languages (Go, Java, Python, Javascript, etc)
    - API and SDK implementation
    - Popular frameworks & libraries instrumentation
    - Collector that receives, processes, and exports data to different backends
    - And of course, auto-instrumentation

@calberto
@codeboten

What's Auto-Instrumentation?

Cross-language requirements for automated approaches to extracting portable telemetry data with zero source code modification.

https://github.com/open-telemetry/oteps/blob/master/text/0001-telemetry-without-manual-instrumentation.md

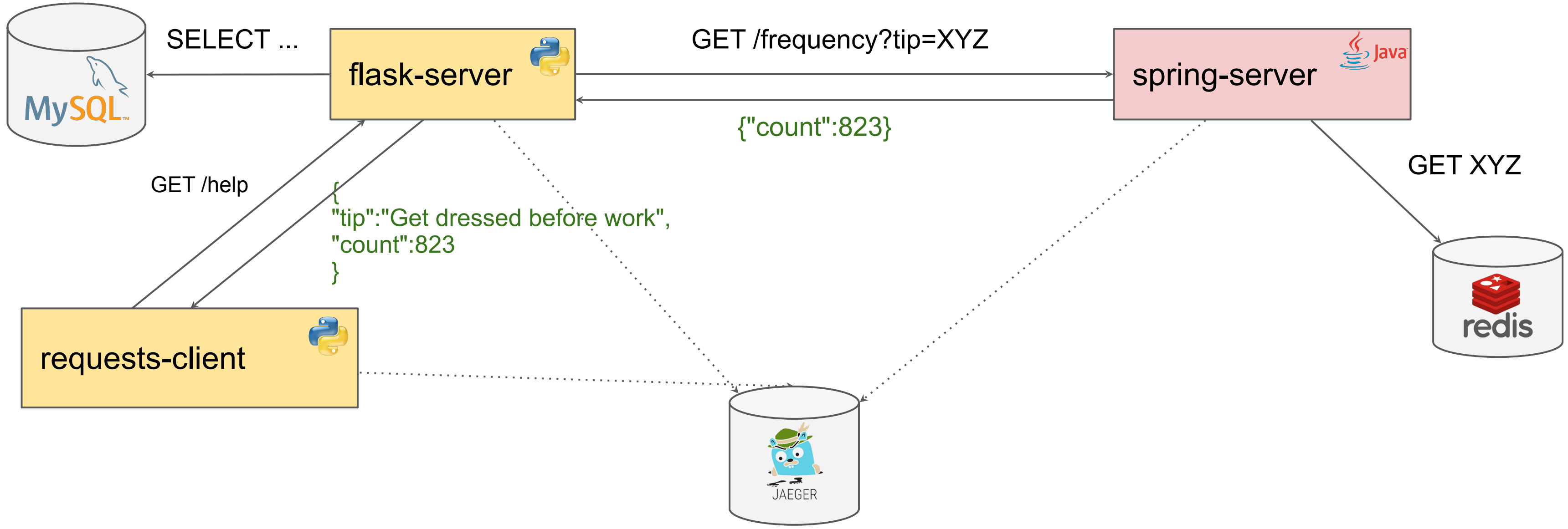


Why Auto-Instrumentation?

- Reduces time to instrument
- Provides insight to libraries

DEMO

But first, a picture



github.com/lightstep/kubecon-otel-auto-instrumentation

@calberto
@codeboten

NOW DEMO

@calberto
@codeboten

How does it work?

Java

@calberto
@codeboten

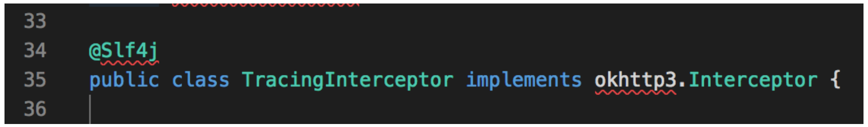
Installing instrumentation

```
wget https://github.com/open-telemetry/opentelemetry-java-instrumentation/releases/latest/download/opentelemetry-javaagent-all.jar
```

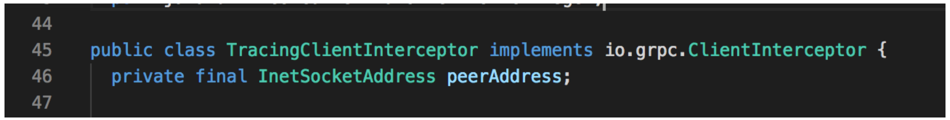
- auto-instrumentation engine
- instrumentation for popular libraries:
  - spring
  - gRPC
  - okhttp
  - etc.
- standard exporters:
  - Jaeger
  - Zipkin
  - **OTLP**

Instrumentation library

- **Instrumentation** uses available hooks, events and interceptor facilities the libraries expose
- **bytebuddy** to do bytecode manipulation



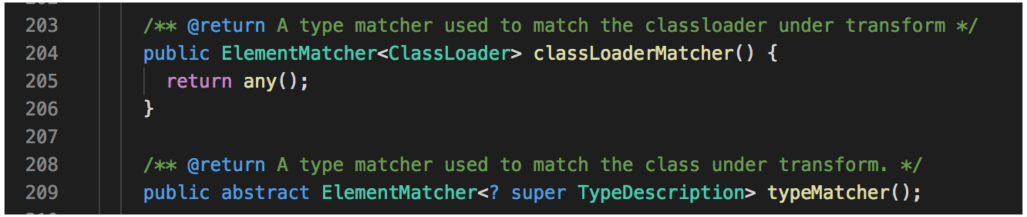
```
@Slf4j
public class TracingInterceptor implements okhttp3.Interceptor {
```



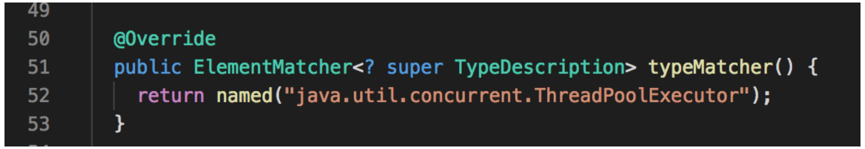
```
public class TracingClientInterceptor implements io.grpc.ClientInterceptor {
  private final InetSocketAddress peerAddress;
```

Instrumentation library

- Implement the Instrumenter interface.
  - Auto detected at runtime



```java
/** @return A type matcher used to match the classloader under transform */
public ElementMatcher<ClassLoader> classLoaderMatcher() {
  return any();
}

/** @return A type matcher used to match the class under transform. */
public abstract ElementMatcher<? super TypeDescription> typeMatcher();
```



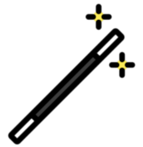
```java
@Override
public ElementMatcher<? super TypeDescription> typeMatcher() {
  return named("java.util.concurrent.ThreadPoolExecutor");
}
```

@calberto
@codeboten

Running the application

- Configuration parameters are passed as Java **system properties** (-D flags) or as **environment variables**.

MAGIC!



```
java -javaagent:opentelemetry-javaagent-all.jar
  -Dota.exporter=otlp
  -Dotel.otlp.endpoint=localhost:55680
  -Dotel.otlp.span.timeout=3000
  -jar myapp.jar
```

```
export OTA_EXPORTER=otlp
export OTA_OTLP_ENDPOINT=localhost:55680
export OTA_OTLP_SPAN_TIMEOUT=3000
java -javaagent:opentelemetry-javaagent-all.jar
  -jar myapp.jar
```

@calberto
@codeboten

How does it work?

Python

@calberto
@codeboten

Installing instrumentation

```
pip install opentelemetry-instrumentation
```

- opentelemetry-bootstrap
- Instrumentor interface
- sitecustomize.py
- opentelemetry-instrument

Installing instrumentation

```
pip install opentelemetry-instrumentation
```

`opentelemetry-bootstrap`

```
root@94d243c69f84:/# opentelemetry-bootstrap
opentelemetry-instrumentation-flask>=0.8b0
opentelemetry-instrumentation-jinja2>=0.8b0
opentelemetry-instrumentation-sqlalchemy>=0.8b0
```

@calberto
@codeboten

Installing instrumentation

```
pip install opentelemetry-instrumentation
```

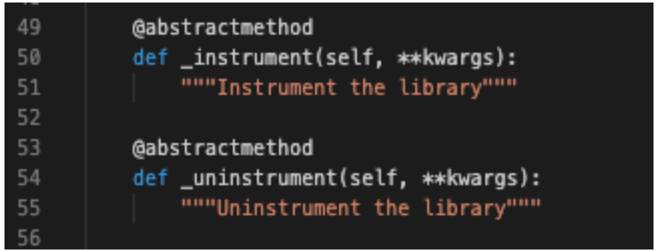
```
opentelemetry-bootstrap
```

`opentelemetry-bootstrap -a install`

```
root@94d243c69f84:/# opentelemetry-bootstrap -a
install
Collecting opentelemetry-instrumentation-
flask>=0.8b0
  Downloading
opentelemetry_instrumentation_flask-0.12b0-py3-
none-any.whl (9.5 kB)
Requirement already satisfied, skipping upgrade:
opentelemetry-api==0.12b0 in /usr/local/lib/
python3.8/site-packages (from opentelemetry-
instrumentation-flask>=0.8b0) (0.12b0)
Collecting opentelemetry-instrumentation-
wsgi==0.12b0
  Downloading
opentelemetry_instrumentation_wsgi-0.12b0-py3-
none-any.whl (9.3 kB)
...
```

Instrumentation library

- Implement the Instrumentor interface

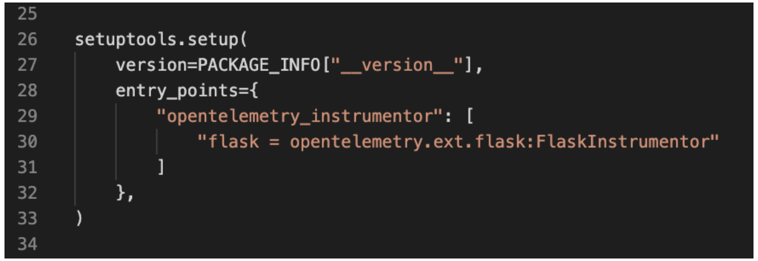


```
    @abstractmethod
    def _instrument(self, **kwargs):
        """Instrument the library"""

    @abstractmethod
    def _uninstrument(self, **kwargs):
        """Uninstrument the library"""

```

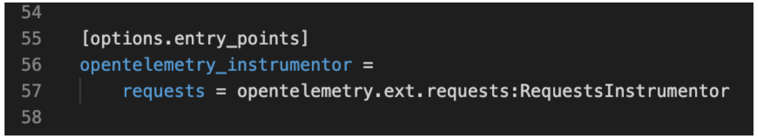
@calberto
@codeboten

Instrumentation library

- Implement the Instrumentor interface
- Register entry point



```
setuptools.setup(
    version=PACKAGE_INFO["__version__"],
    entry_points={
        "opentelemetry_instrumentor": [
            "flask = opentelemetry.ext.flask:FlaskInstrumentor"
        ]
    },
)
```



```
[options.entry_points]
opentelemetry_instrumentor =
    requests = opentelemetry.ext.requests:RequestsInstrumentor
```

@calberto
@codeboten

Running the application

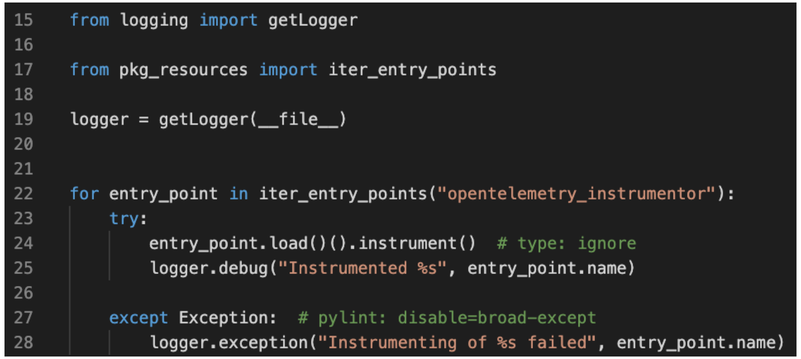
```
opentelemetry-instrument /app/server.py
```

MAGIC!

@calberto
@codeboten

Running the application

- Instrument all libraries

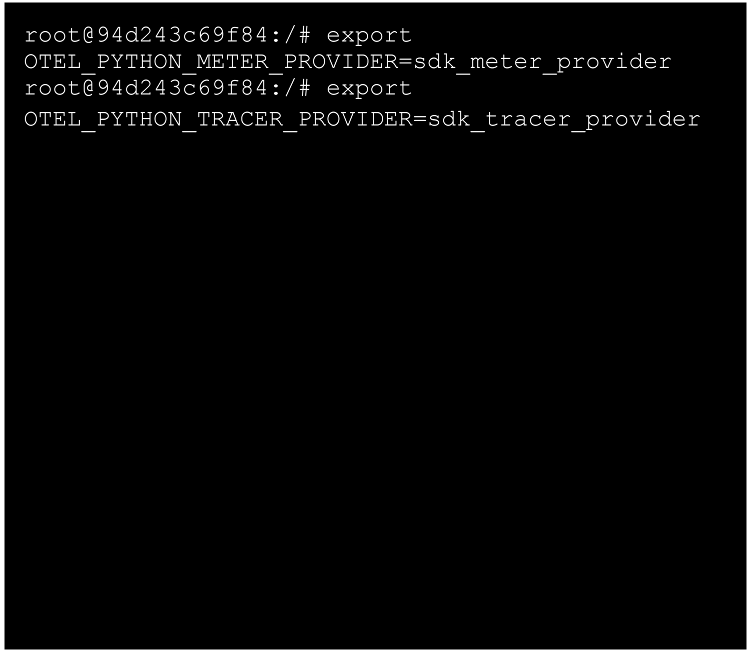


```python
from logging import getLogger

from pkg_resources import iter_entry_points

logger = getLogger(__file__)


for entry_point in iter_entry_points("opentelemetry_instrumentor"):
    try:
        entry_point.load()().instrument()  # type: ignore
        logger.debug("Instrumented %s", entry_point.name)

    except Exception:  # pylint: disable=broad-except
        logger.exception("Instrumenting of %s failed", entry_point.name)
```

@calberto
@codeboten

Gotchas

- Things to configure



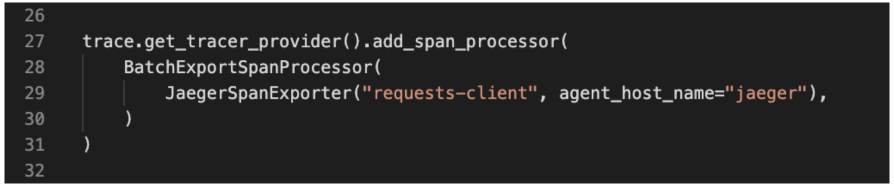
```
root@94d243c69f84:/# export
OTEL_PYTHON_METER_PROVIDER=sdk_meter_provider
root@94d243c69f84:/# export
OTEL_PYTHON_TRACER_PROVIDER=sdk_tracer_provider
```

@calberto
@codeboten

Gotchas

- Things to configure

- Not everything is configurable via environment variables yet!



```
trace.get_tracer_provider().add_span_processor(
    BatchExportSpanProcessor(
        JaegerSpanExporter("requests-client", agent_host_name="jaeger"),
    )
)
```

What's next for instrumentation

@calberto
@codeboten

Instrumentation today

Go	Instrumented libraries
Java	Auto-Instrumentation
Javascript	Auto-Instrumentation
.Net	Under way
Python	Auto-Instrumentation
Ruby	Auto-Instrumentation

Next steps

- More work
  - many languages
  - many libraries

- Get involved
  - try existing libraries
  - open an issue for your favourite library/framework
  - send PRs for code and docs!!

More info

https://github.com/lightstep/kubecon-otel-auto-instrumentation

https://opentelemetry.io

https://opentelemetry.io/registry

Questions

calberto

carlosalberto

codeboten

codeboten

KubeCon | CloudNativeCon

Europe 2020

Virtual

KEEP CLOUD NATIVE

CONNECTED