

A stylized graphic of a telescope, composed of several colored segments: a yellow eyepiece, a blue barrel, a blue objective lens housing, and a yellow objective lens. The telescope is angled upwards from the bottom left towards the top right.

OpenTelemetry

Ashish Tiwari
Senior Developer Advocate



Monitoring \neq Observability

Monitoring \neq Observability

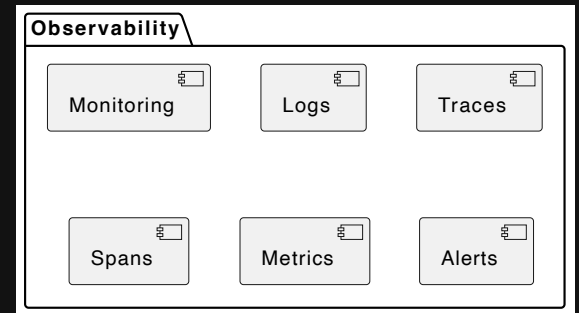
- When & What
 - Monitor
 - Watch
 - Trigger
 - Alert

Monitoring \neq Observability

- When & What
 - Monitor
 - Watch
 - Trigger
 - Alert
- Why & How
 - RCA
 - Detection
 - Correlations
 - Anomaly detection

Monitoring \neq Observability

- When & What
 - Monitor
 - Watch
 - Trigger
 - Alert
- Why & How
 - RCA
 - Detection
 - Correlations
 - Anomaly detection



Performance that Delivers Relevant Results in Real-time







OpenTelemetry is an Observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs.



OpenTelemetry is an Observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs.

- Why OTel?



OpenTelemetry is an Observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs.

- Why OTel?
 - No Vendor lock-in



OpenTelemetry is an Observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs.

- Why OTel?
 - No Vendor lock-in
 - Single set of APIs and conventions



OpenTelemetry is an Observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs.

- Why OTel?
 - No Vendor lock-in
 - Single set of APIs and conventions
 - Easy switch among backend



OpenTelemetry is an Observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs.

- Why OTel?
 - No Vendor lock-in
 - Single set of APIs and conventions
 - Easy switch among backend
- Instrumentation



OpenTelemetry is an Observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs.

- Why OTel?
 - No Vendor lock-in
 - Single set of APIs and conventions
 - Easy switch among backend
- Instrumentation
 - Zero code (auto): Bytecode, monkey patching, eBPF etc.



OpenTelemetry is an Observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs.

- Why OTel?
 - No Vendor lock-in
 - Single set of APIs and conventions
 - Easy switch among backend
- Instrumentation
 - Zero code (auto): Bytecode, monkey patching, eBPF etc.
 - Code-based (manual): OTel API & SDKs



OpenTelemetry is an Observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs.

- Why OTel?
 - No Vendor lock-in
 - Single set of APIs and conventions
 - Easy switch among backend
- Instrumentation
 - Zero code (auto): Bytecode, monkey patching, eBPF etc.
 - Code-based (manual): OTel API & SDKs
 - Libraries and Frameworks



OpenTelemetry is an Observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs.

- Why OTel?

- No Vendor lock-in
- Single set of APIs and conventions
- Easy switch among backend

- Instrumentation

- Zero code (auto): Bytecode, monkey patching, eBPF etc.
- Code-based (manual): OTel API & SDKs
- Libraries and Frameworks

- Language APIs & SDKs





OpenTelemetry is an Observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs.

- Why OTel?

- No Vendor lock-in
- Single set of APIs and conventions
- Easy switch among backend

- Instrumentation

- Zero code (auto): Bytecode, monkey patching, eBPF etc.
- Code-based (manual): OTel API & SDKs
- Libraries and Frameworks

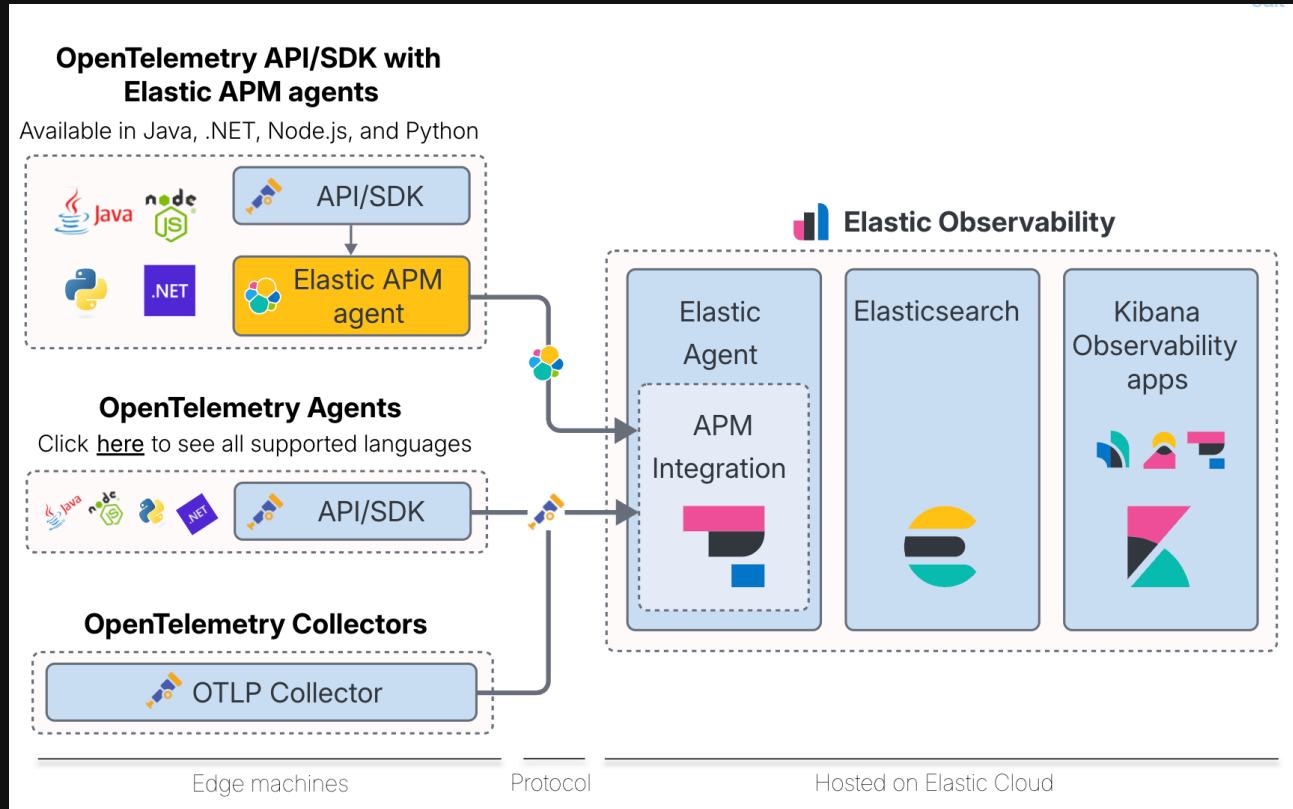
- Language APIs & SDKs



Checkout the registry to know about instrumentation libraries, collector components, utilities, and other useful projects in the OpenTelemetry ecosystem.

OTel -> Elastic

Get telemetry data into Elastic



Let's see how it works with Flask - Python web framework

Flask is a micro web framework written in Python. It use to build lightweight web applications quickly.

```
pip install flask
```

```
# install dependency
pip install opentelemetry-distro
opentelemetry-bootstrap -a install
```

```
# run flask app
flask run -p 8080
```

```
# run instrumented flask app
export OTEL_PYTHON_LOGGING_AUTO_INSTRUMENTATION_ENABLED=true
opentelemetry-instrument \
  --traces_exporter console \
  --metrics_exporter console \
  --logs_exporter console \
  --service_name dice-server \
  flask run -p 8080
```

app.py

```
from flask import Flask
app = Flask(__name__)

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

<https://opentelemetry.io/docs/languages/python/>

Let's see how it works with Flask - Python web framework

Flask is a micro web framework written in Python. It use to build lightweight web applications quickly.

```
pip install flask
```

```
# install dependency
pip install opentelemetry-distro
opentelemetry-bootstrap -a install
```

```
# run flask app
flask run -p 8080
```

```
# run instrumented flask app
export OTEL_PYTHON_LOGGING_AUTO_INSTRUMENTATION_ENABLED=true
opentelemetry-instrument \
  --traces_exporter console \
  --metrics_exporter console \
  --logs_exporter console \
  --service_name dice-server \
  flask run -p 8080
```

app.py

```
from flask import Flask
app = Flask(__name__)

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

<https://opentelemetry.io/docs/languages/python/>

Let's see how it works with Flask - Python web framework

Flask is a micro web framework written in Python. It use to build lightweight web applications quickly.

```
pip install flask
```

```
# install dependency
pip install opentelemetry-distro
opentelemetry-bootstrap -a install
```

```
# run flask app
flask run -p 8080
```

```
# run instrumented flask app
export OTEL_PYTHON_LOGGING_AUTO_INSTRUMENTATION_ENABLED=true
opentelemetry-instrument \
  --traces_exporter console \
  --metrics_exporter console \
  --logs_exporter console \
  --service_name dice-server \
  flask run -p 8080
```

app.py

```
from flask import Flask
app = Flask(__name__)

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

<https://opentelemetry.io/docs/languages/python/>

Let's see how it works with Flask - Python web framework

Flask is a micro web framework written in Python. It use to build lightweight web applications quickly.

```
pip install flask
```

```
# install dependency
pip install opentelemetry-distro
opentelemetry-bootstrap -a install
```

```
# run flask app
flask run -p 8080
```

```
# run instrumented flask app
export OTEL_PYTHON_LOGGING_AUTO_INSTRUMENTATION_ENABLED=true
opentelemetry-instrument \
  --traces_exporter console \
  --metrics_exporter console \
  --logs_exporter console \
  --service_name dice-server \
  flask run -p 8080
```

app.py

```
from flask import Flask
app = Flask(__name__)

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

<https://opentelemetry.io/docs/languages/python/>

Demo - ElasticFlix

<https://github.com/elastic/observability-examples/tree/main/Elastiflix>

Get started with OTEL + Elastic

Independence with
OpenTelemetry on Elastic

<https://www.elastic.co/observability-labs/blog/opentelemetry-observability>

What is OTEL

<https://www.elastic.co/what-is/opentelemetry>

Semantic Conventions for
Elasticsearch

<https://opentelemetry.io/docs/specs/semconv/database/elasticsearch/>

Instrumentation

<https://opentelemetry.io/docs/concepts/instrumentation/>

Other concepts


Baggage, Context Propagation

OTEL

<https://opentelemetry.io>

Thank You

 [in/ashishtiwari93](https://www.linkedin.com/in/ashishtiwari93)

 [@_ashish_tiwari](https://twitter.com/_ashish_tiwari)