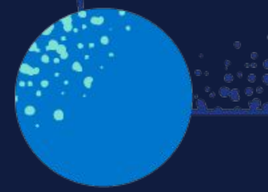
Introduction to OpenTelemetry with Python



elastic

Jessica Garson

@JessicaGarson

@JessicaGarson@macaw.social

Senior Developer Advocate

elastic

https://github.com/JessicaGarson/Introduction-to-OpenTelemetry-with-Python

elastic

Agenda

- Observability basics
- What is OpenTelemetry and why use it
- Django and Flask instrumentation
- The state of OpenTelemetry
- Conclusion and next steps

I had attempted to find a short primer on starting with OpenTelemetry with Python but couldn’t find much more than a **demo application**.

https://opentelemetry-python.readthedocs.io/en/latest/examples/django/README.html
https://opentelemetry.io/docs/languages/python/getting-started/

elastic

ChatGPT

Sure! Imagine you have a big box of crayons (these are like different parts of a computer program) and you want to make a beautiful picture (which is like running the program to do something cool). But sometimes, while you're drawing, a crayon might break or not color the way it should. This can make your picture not turn out as nice as you wanted.

OpenTelemetry is like having a magical friend who watches you while you draw. This friend takes notes on which crayons you use, how they perform, and if they break. Then, if something goes wrong, your magical friend can tell you exactly which crayon broke and why, so you can fix it and make sure your picture turns out great next time!

So, OpenTelemetry helps people who make computer programs make sure their programs run smoothly and fix any problems that happen.


elastic

Observability fundamentals

elastic

Observability helps you determine what's happening inside your system by analyzing the data it outputs

elastic

elastic

What isn’t operating as expected

elastic

Beyond detecting problems

elastic

What is an agent?

elastic

An observability agent is a software component or tool that collects, processes, and sends telemetry data from a system or application to an observability platform for analysis.

elastic

Logs

Logs are records of events in a system, documenting operations, errors, and activities to aid in troubleshooting, monitoring, and compliance.

elastic

Metrics

Metrics are quantitative measurements that track the performance and health of a system

Traces

Traces track the path and interactions of a request through a system.

What is OpenTelemetry (OTel)?

OpenTelemetry (OTel) is an observability framework that is open source and vendor neutral, designed to work with any backend system.

It provides standardized APIs, libraries, and tools to collect telemetry data such as metrics, logs, and traces.

OpenTelemetary enables comprehensive monitoring across services and platforms.

elastic

Why?


Logs alone are simply not
enough sometimes

Um, what?

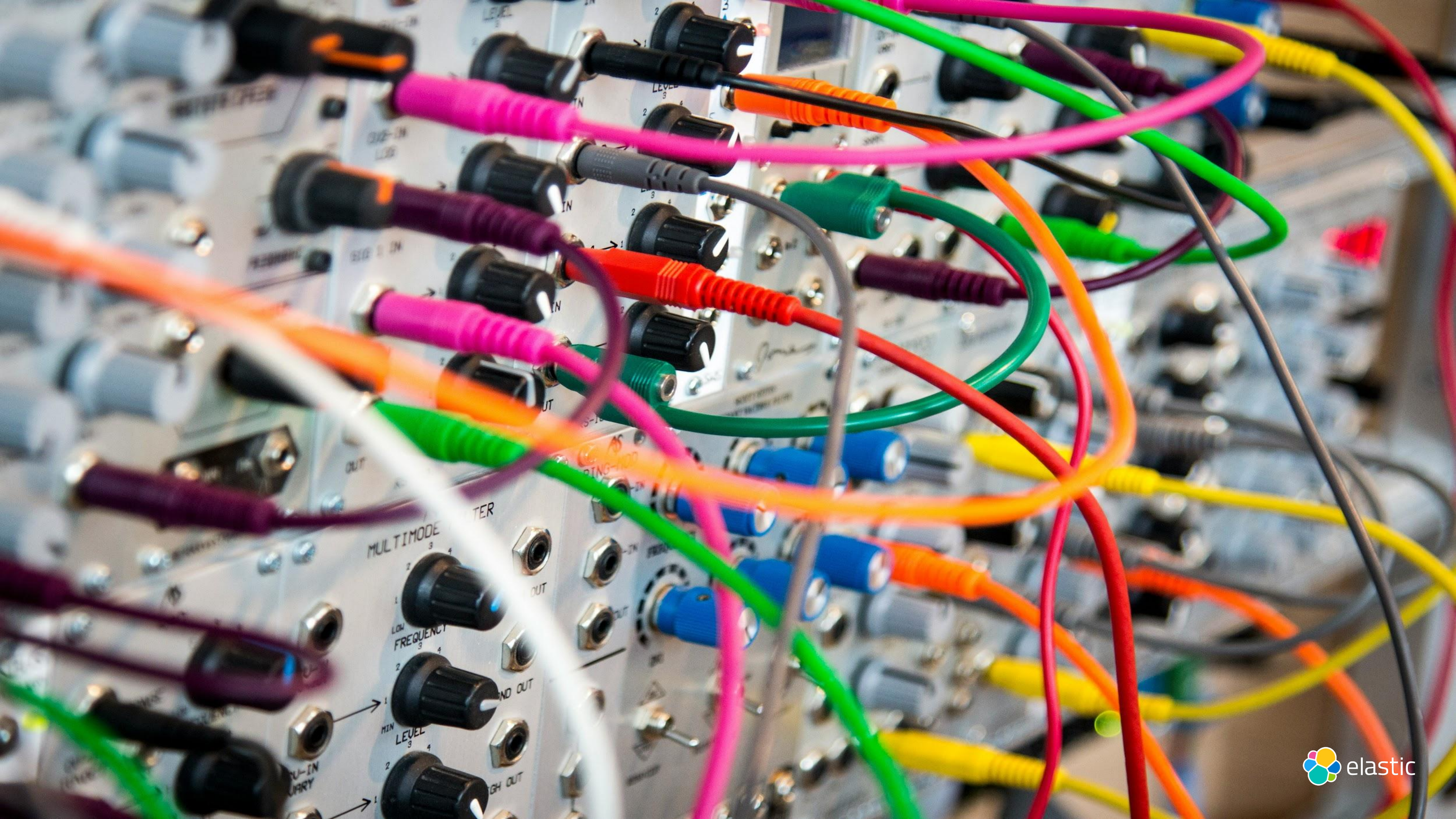
```
> * 2024-05-10 08:00:06.117  55  54440319  ...  http://api.nasa.gov/neo/rest/v1/neo/sentry/544...
> * 2024-05-10 08:00:06.117  56  54440642  ...                                                NaN
v * 2024-05-10 08:00:06.117  57  54440644  ...  http://api.nasa.gov/neo/rest/v1/neo/sentry/544...
```

Copy ▾ | Similar entries ▾ | Expand nested fields | Hide log summary


```
{
  insertId: "663e0c460001c9a60e5e65c3"
  labels: {2}
  logName: "projects/ncav-293119/logs/run.googleapis.com%2Fstdout"
  receiveTimestamp: "2024-05-10T12:00:06.338017672Z"
  resource: {2}
  textPayload: "57  54440644  ...  http://api.nasa.gov/neo/rest/v1/neo/sentry/544..."
  timestamp: "2024-05-10T12:00:06.117158Z"
}
```


```
> * 2024-05-10 08:00:06.117  [43 rows x 18 columns]
```

elastic


MULTIMODE
FREQUENCY
LEVEL
OUT
elastic

Future proof

Spans within traces? What?!?!

elastic

Spans are the building blocks of traces.
They include the following information:

- Name
- Parent span ID (empty for root spans)

elastic

```
{
    "name": "add_item_view_span",
    "context": {
        "trace_id": "0x81364006fe668cb26a54e20918be9620",
        "span_id": "0x9a048a860bb7c9ac",
        "trace_state": "[]"
    },
    "kind": "SpanKind.INTERNAL",
    "parent_id": null,
    "start_time": "2024-08-15T20:55:42.634538Z",
    "end_time": "2024-08-15T20:55:42.637830Z",
    "status": {
        "status_code": "UNSET"
    },
    "attributes": {},
    "events": [],
    "links": [],
    "resource": {
        "attributes": {
            "service.name": "to-do-list-app-manual",
            "service.version": "1.0.0"
        },
        "schema_url": ""
    }
}
```


OHHH!

A span within a trace!

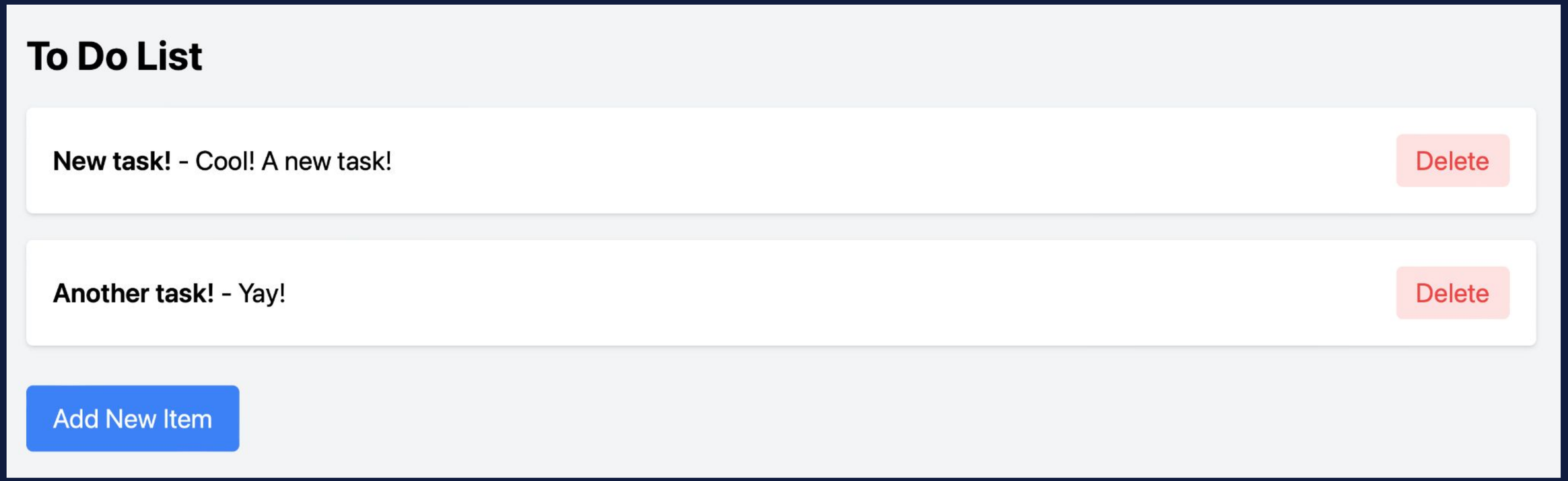
elastic

Instrumenting refers to the process of adding observability features to your application to collect telemetry data, such as traces, metrics, and logs.

elastic

Django Example

The classic Django example



elastic

How do you get that data to show in your console?

elastic

```
pip install django django-environ elastic-opentelemetry opentelemetry-instrumentation-django
opentelemetry-bootstrap --action=install
```

elastic

settings.py

```
env = environ.Env()

environ.Env.read_env(os.path.join(BASE_DIR, '.env'))

OTEL_EXPORTER_OTLP_HEADERS = env('OTEL_EXPORTER_OTLP_HEADERS')
OTEL_EXPORTER_OTLP_ENDPOINT = env('OTEL_EXPORTER_OTLP_ENDPOINT')
```

elastic

.env example

```
OTEL_EXPORTER_OTLP_HEADERS="Authorization=ApiKey%20yourapikey"
OTEL_EXPORTER_OTLP_ENDPOINT="https://your/host/endpoint"
```

elastic

```python
import os
import sys
from opentelemetry import trace
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.sdk.trace.export import BatchSpanProcessor, ConsoleSpanExporter
from opentelemetry.sdk import resources

def main():
    """Run administrative tasks."""
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "todolist_project.settings")

    resource = resources.Resource(attributes={
        resources.SERVICE_NAME: "your-service-name",
        resources.SERVICE_VERSION: "1.0.0"
    })

    trace_provider = TracerProvider(resource=resource)
    trace.set_tracer_provider(trace_provider)

    console_exporter = ConsoleSpanExporter()

    span_processor = BatchSpanProcessor(console_exporter)
    trace.get_tracer_provider().add_span_processor(span_processor)
```

manage.py

elastic

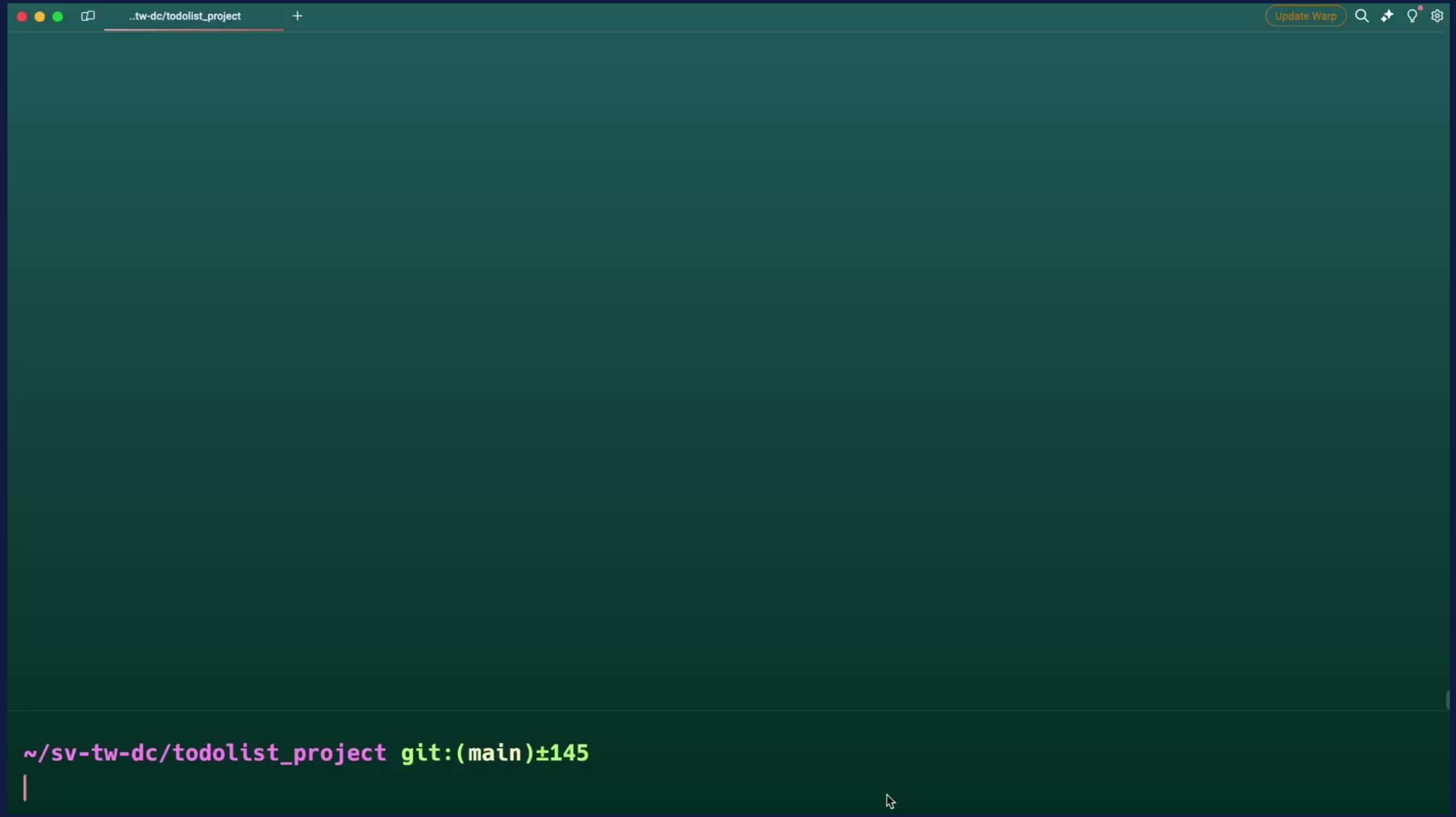
Automatic Instrumentation

The process by which an agent modifies the bytecode of your application's classes, often to insert monitoring code

manage.py

```
def main():
    """Run administrative tasks."""
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "todolist_project.settings")
    DjangoInstrumentor().instrument()

    # Set up resource attributes for the service
    resource = resources.Resource(attributes={
        resources.SERVICE_NAME: "to-do-list-app",
        resources.SERVICE_VERSION: "1.0.0"
    })

    trace_provider = TracerProvider(resource=resource)
    trace.set_tracer_provider(trace_provider)

    otlp_exporter = OTLPSpanExporter()

    # Set up the BatchSpanProcessor to export traces
    span_processor = BatchSpanProcessor(otlp_exporter)
    trace.get_tracer_provider().add_span_processor(span_processor)
    turn go(f, seed, [])
}
```

elastic


..tw-dc/todolist_project
Update Warp
~/sv-tw-dc/todolist_project git:(main)±145

elastic

Manual instrumentation

Manual instrumentation requires incorporating particular code segments into your application to collect and transmit telemetry data.

So when would you want to use manual instrumentation?

Step 1: Delete this line from your manage.py file

```
# DjangoInstrumentor().instrument()
```

elastic

Step 2: Update your views.py

```
tracer = trace.get_tracer(__name__)

meter = get_meter(__name__)

view_counter = meter.create_counter(
    "view_requests",
    description="Counts the number of requests to views",
)

view_duration_histogram = meter.create_histogram(
    "view_duration",
    description="Measures the duration of view execution",
)
```

```python
def index(request):
    start_time = time()

    with tracer.start_as_current_span("index_view_span") as span:
        items = ToDoItem.objects.all()
        span.set_attribute("todo.item_count", items.count())
        response = render(request, 'todo/index.html', {'items': items})

    view_counter.add(1, {"view_name": "index"})
    view_duration_histogram.record(time() - start_time, {"view_name": "index"})

    return response
```

```python
def add_item(request):
    start_time = time()

    with tracer.start_as_current_span("add_item_view_span") as span:
        if request.method == 'POST':
            form = ToDoForm(request.POST)
            if form.is_valid():
                form.save()
                span.add_event("New item added")
                response = redirect('index')
            else:
                response = render(request, 'todo/add_item.html', {'form': form})
        else:
            form = ToDoForm()
            response = render(request, 'todo/add_item.html', {'form': form})

    view_counter.add(1, {"view_name": "add_item"})
    view_duration_histogram.record(time() - start_time, {"view_name": "add_item"})

    return response
```

```python
def delete_item(request, item_id):
    start_time = time()

    with tracer.start_as_current_span("delete_item_view_span") as span:
        item = get_object_or_404(ToDoItem, id=item_id)
        span.set_attribute("todo.item_id", item_id)
        if request.method == 'POST':
            item.delete()
            span.add_event("Item deleted")
            response = redirect('index')
        else:
            response = render(request, 'todo/delete_item.html', {'item': item})

    view_counter.add(1, {"view_name": "delete_item"})
    view_duration_histogram.record(time() - start_time, {"view_name": "delete_item"})

    return response
```

Step 3: Update your models.py

```python
from django.db import models
from opentelemetry import trace

tracer = trace.get_tracer(__name__)

class ToDoItem(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField(blank=True)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title

    def save(self, *args, **kwargs):
        with tracer.start_as_current_span("save_todo_item_span") as span:
            span.set_attribute("todo.title", self.title)
            if self.pk:
                span.add_event("Updating ToDoItem")
            else:
                span.add_event("Creating new ToDoItem")

            super(ToDoItem, self).save(*args, **kwargs)
```

elastic

```
python manage.py runserver

Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 23, 2024 - 23:36:52
Django version 5.1, using settings 'todolist_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

[23/Oct/2024 23:36:59] "GET / HTTP/1.1" 200 1695
```

Flask

No code instrumentation for automatic instrumentation

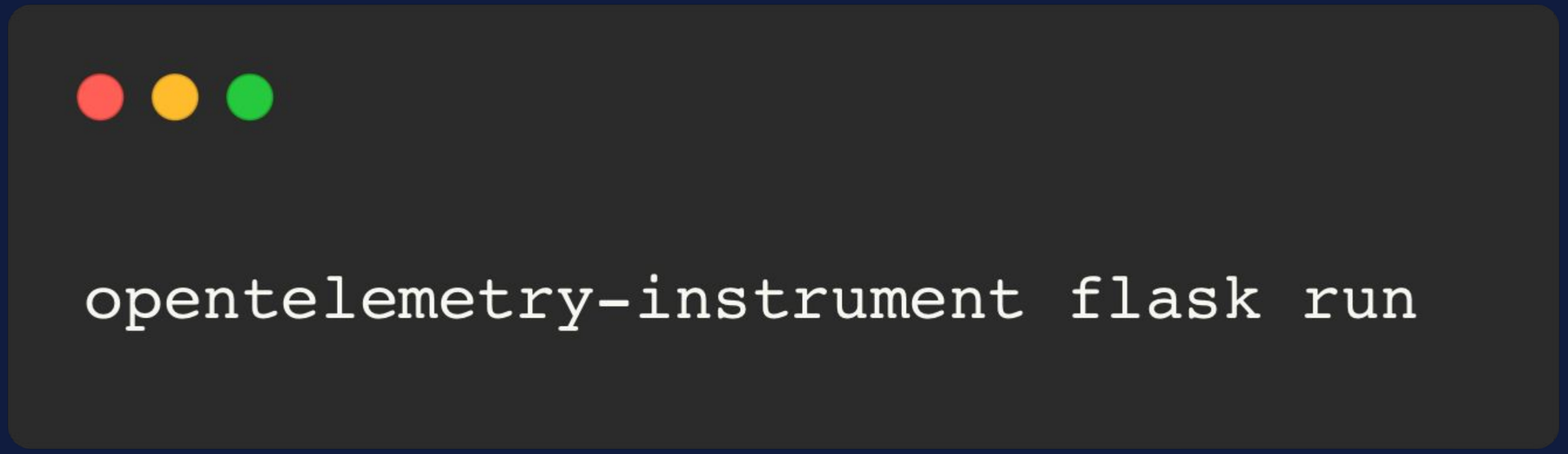
elastic

elastic

elastic

```
export OTEL_RESOURCE_ATTRIBUTES="service.name=flask-todo-app"
export OTEL_EXPORTER_OTLP_HEADERS="Authorization=<your_authorization_header_value>"
export OTEL_EXPORTER_OTLP_ENDPOINT="<your_elastic_cloud_url>"
```

elastic



elastic

To-Do List

Add new task

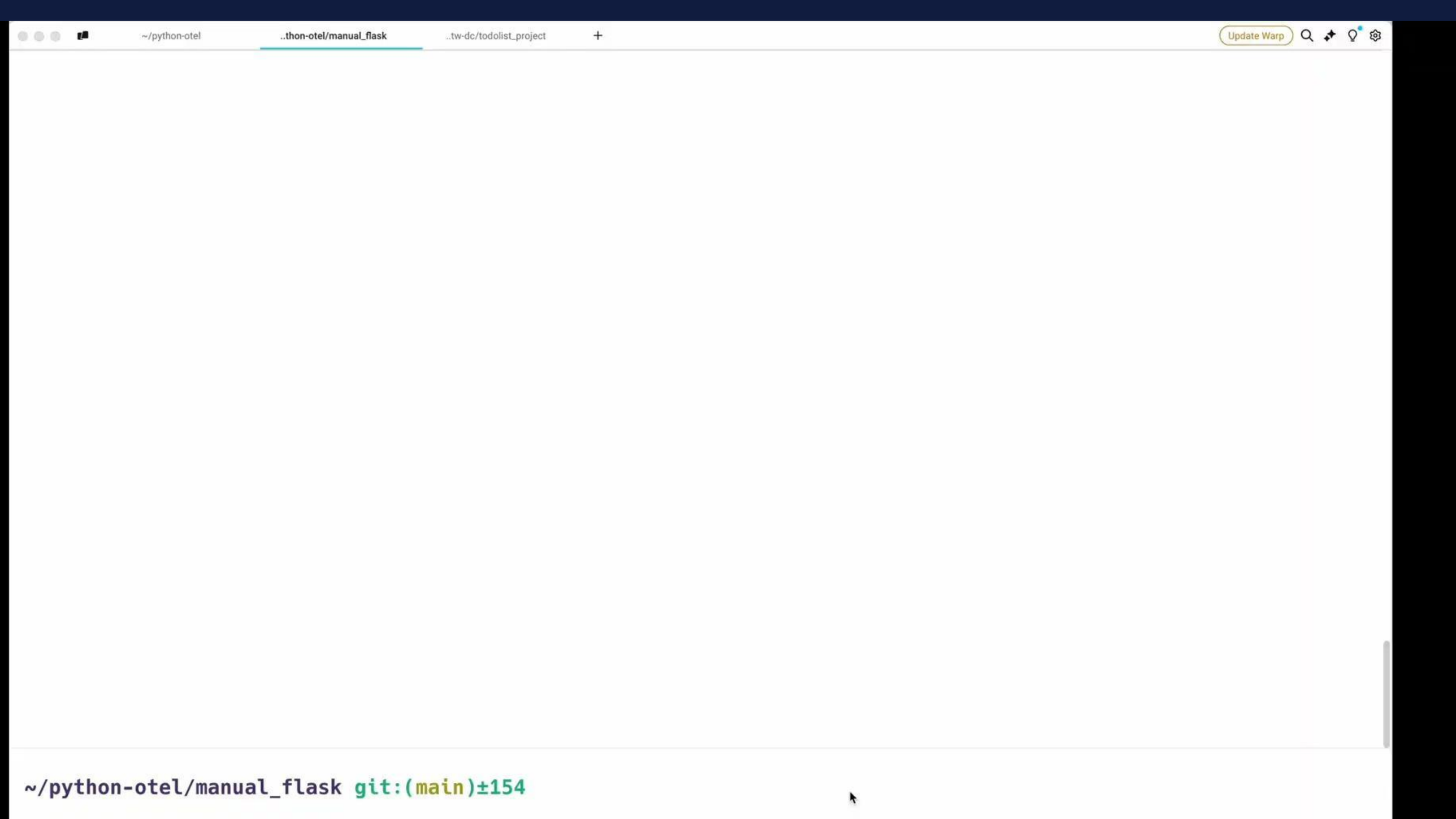
ADD TASK

New task! Delete

elastic

Manual instrumentation in Flask

elastic

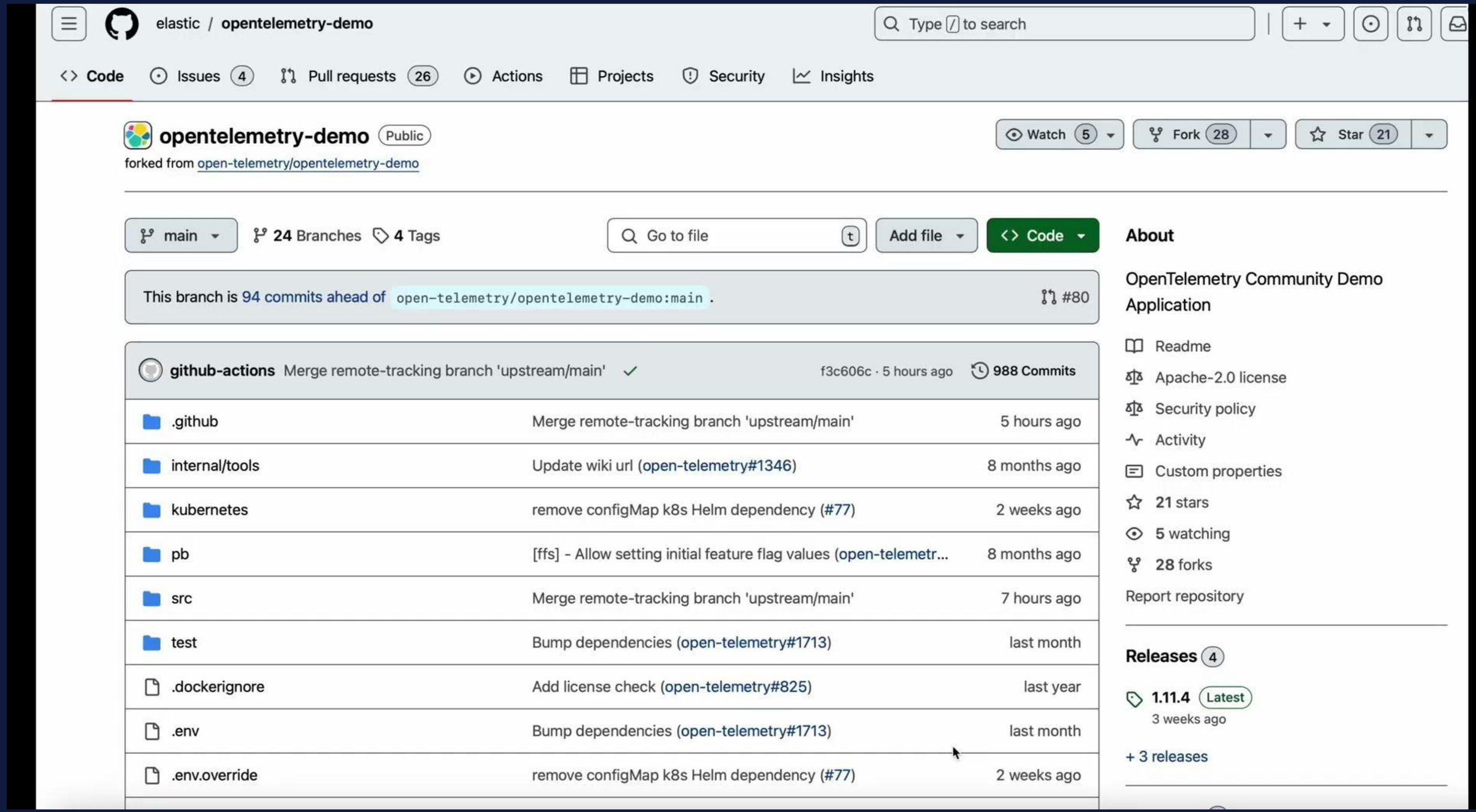

~/python-otel
..thon-otel/manual_flask
..tw-dc/todolist_project
Update Warp
~/python-otel/manual_flask git:(main)±154

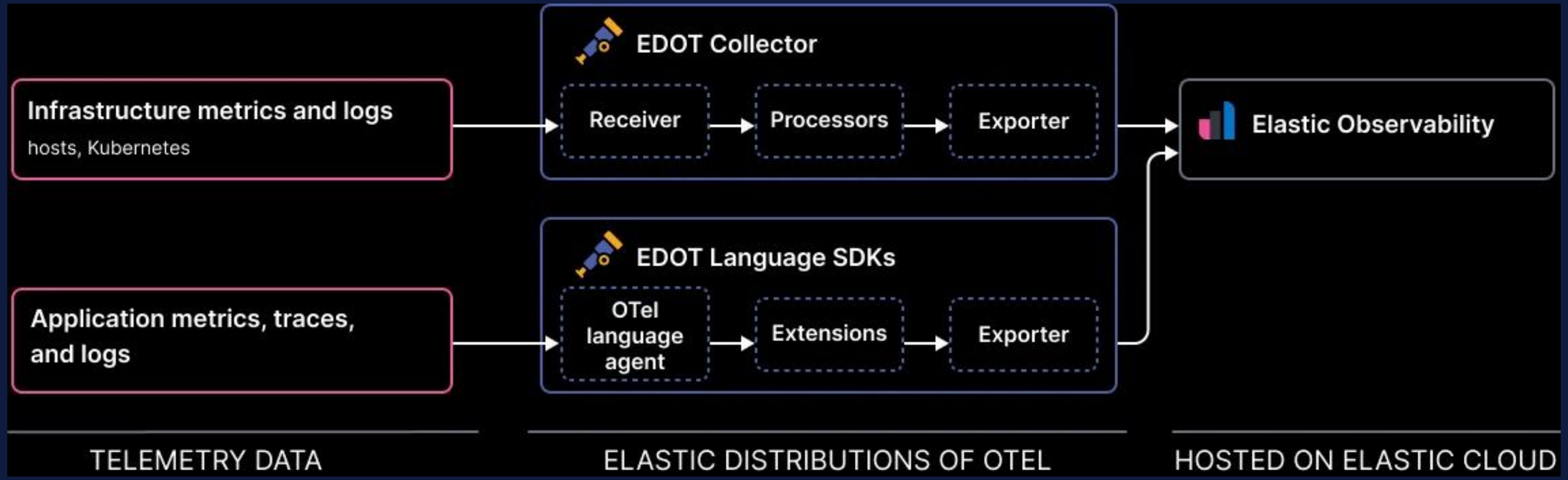
OpenTelemetry Collector

A solution for receiving, processing, and exporting telemetry data. It eliminates the need to manage and maintain multiple agents or collectors.

https://opentelemetry.io/docs/demo

elastic


elastic / opentelemetry-demo
Type / to search
Code
Issues 4
Pull requests 26
Actions
Projects
Security
Insights
opentelemetry-demo
Public
forked from open-telemetry/opentelemetry-demo
Watch 5
Fork 28
Star 21
main
24 Branches
4 Tags
Go to file
t
Add file
Code
This branch is 94 commits ahead of open-telemetry/opentelemetry-demo:main .
#80
github-actions Merge remote-tracking branch 'upstream/main'
f3c606c · 5 hours ago
988 Commits
.github Merge remote-tracking branch 'upstream/main' 5 hours ago
internal/tools Update wiki url (open-telemetry#1346) 8 months ago
kubernetes remove configMap k8s Helm dependency (#77) 2 weeks ago
pb [ffs] - Allow setting initial feature flag values (open-telemetr... 8 months ago
src Merge remote-tracking branch 'upstream/main' 7 hours ago
test Bump dependencies (open-telemetry#1713) last month
.dockerignore Add license check (open-telemetry#825) last year
.env Bump dependencies (open-telemetry#1713) last month
.env.override remove configMap k8s Helm dependency (#77) 2 weeks ago
About
OpenTelemetry Community Demo Application
Readme
Apache-2.0 license
Security policy
Activity
Custom properties
21 stars
5 watching
28 forks
Report repository
Releases 4
1.11.4 Latest
3 weeks ago
+ 3 releases


EDOT Collector
Infrastructure metrics and logs
hosts, Kubernetes
Receiver
Processors
Exporter
Elastic Observability
EDOT Language SDKs
Application metrics, traces, and logs
OTel language agent
Extensions
Exporter
TELEMETRY DATA
ELASTIC DISTRIBUTIONS OF OTEL
HOSTED ON ELASTIC CLOUD

elastic

State of adoption

elastic

https://opentelemetry.io/status/

Closing thoughts

elastic

OpenTelemetry (OTel) is an open-source, vendor-neutral observability framework designed to integrate with any backend system.

elastic

It's very easy to start instrumenting your code.

It is highly configurable and extensible

It scales well from large applications to small applications

elastic

Let me know if this talk inspires you to build anything. I'm @JessicaGarson on most platforms.


Thank you!
elastic