

pydantic

LOGFIRE

Why?

Building Libraries is easy

Building apps is much harder

- Networks
- Databases
- APIs
- Infra
- Users

I had a feeling the missing link was logging(ish)

What's a log?

A 1D list of tuple:

```
list[tuple[datetime, Level, str]]
```

```
[  
  (datetime(...), 'info',      'GET /accounts/invoice/ -> 200'),  
  (datetime(...), 'info',      'GET /add-on/complete/ -> 200'),  
  (datetime(...), 'error',     'Stripe payment failed, insuff...'),  
  (datetime(...), 'warning',   'POST /payments/42/complete/ -> 400'),  
  (datetime(...), 'warning',   'POST /payments/37/complete/ -> 400'),  
  (datetime(...), 'debug',     'Slow query detected SELECT...'),  
]
```



This is what code looked like...
until about 1980

How it should be

A nested structure:

```
[
  Log(
    start=datetime(...),
    end=datetime(...),
    message='POST /payments/42/complete/ -> 400',
    children=[
      Log(
        start=datetime(...),
        message='Stripe payment failed, insuff...',
        attributes={'reason': 'insufficient funds', ...}
      ),
    ]
  )
]
```

What we proposing here isn't new, but it's
still not available to most developers

So we built it

```
1 @app.post('/payments/{user_id:int}/complete/')

```

🔗 samuelcolvin > foobar2

Live

Dashboards

Alerts

Explore

Settings

🔍 Feedback / Help

📄 Beta



Enter SQL to filter...



May 16, 07:26:41 07:27:31 07:28:21 07:29:11 07:30:01 07:30:51 May 16, 07:31:41



Connected

Visibility ▾

Default levels ▾

Last 5 minutes ▾

Showing all traces after 2024-05-16 07:26:41

07:29:21	◆ Starting the app	logfire	
07:29:58	◆ Starting the app	logfire	
07:30:04	4-- POST /payments/42/complete/	asgi	1.6s
07:30:04	◆ FastAPI arguments	fastapi logfire	0.0ms
07:30:04	◆ Calling demo.get_payment_details	logfire	206ms
07:30:04	◆ stripe payment amount=2000 currency='usd'	logfire	1.2s
07:30:05	◆ Calling demo.store_payment_success	logfire	202ms
07:30:32	4-- POST /payments/37/complete/	asgi	1.4s
07:30:32	◆ FastAPI arguments	fastapi logfire	1ms
07:30:32	◆ Calling demo.get_payment_details	logfire	205ms
07:30:32	◆ stripe payment amount=2000 currency='usd'	logfire	1.0s
07:30:33	◆ Calling demo.store_payment_failure	logfire	205ms



Span: stripe payment amount=2000
currency='usd' user_id=42



Service Name unknown_service

Trace ID ↔ #89a5629b71ab624836811477a932022d

Span ID ↔ #de9af70c114669e7 Date 2024-05-16

Time span 07:30:04.449 to 07:30:05.644 Duration 1.2s

Details Raw Data

Arguments: (as Python)

```
✓ { 4 items
  'amount': 2000,
  'user_id': 42,
  'currency': 'usd',
  'payment_intent': { 40 items
    'id': 'pi_3PH2VcB7axLxkstu0N4eFnX0',
    'amount': 2000,
```

C'mon, that's not new

No, but these are:

- Using OpenTelemetry ... Taming OpenTelemetry
- Python magic:
 - Autotracing - aka profiling
 - f-string magic
- GenAI/LLM + general purpose
- SQL
- But most of all: **it's easy to use**

**We're the right people to build
this **because** we're not
observability professionals**

but, we need your help to make it great

Enough theory

Show me the code!

Manual tracing

May 16, 09:47:5809:48:4809:49:3809:50:2809:51:1809:52:08May 16, 09:52:58

Connected

Visibility ▾ Default levels ▾ Last 5 minutes ▾

Showing all traces after 2024-05-16 09:47:58

09:52:23 ◆ Hello world

09:52:23 2- doing some slow work... 590ms

09:52:23 ◆ fruit=banana

09:52:23 2- more nesting 463ms

09:52:23 ◆ this is ominous

09:52:24 ◆ done

Log: fruit=banana

Service Name unknown_service

Trace ID ↔ #a26b95b77b9fa56e77889f67ac9e6bd9

Span ID ↔ #614fa6622aeb02c8

Timestamp 2024-05-16 09:52:23.569

Details Raw Data

Arguments: (as Python)

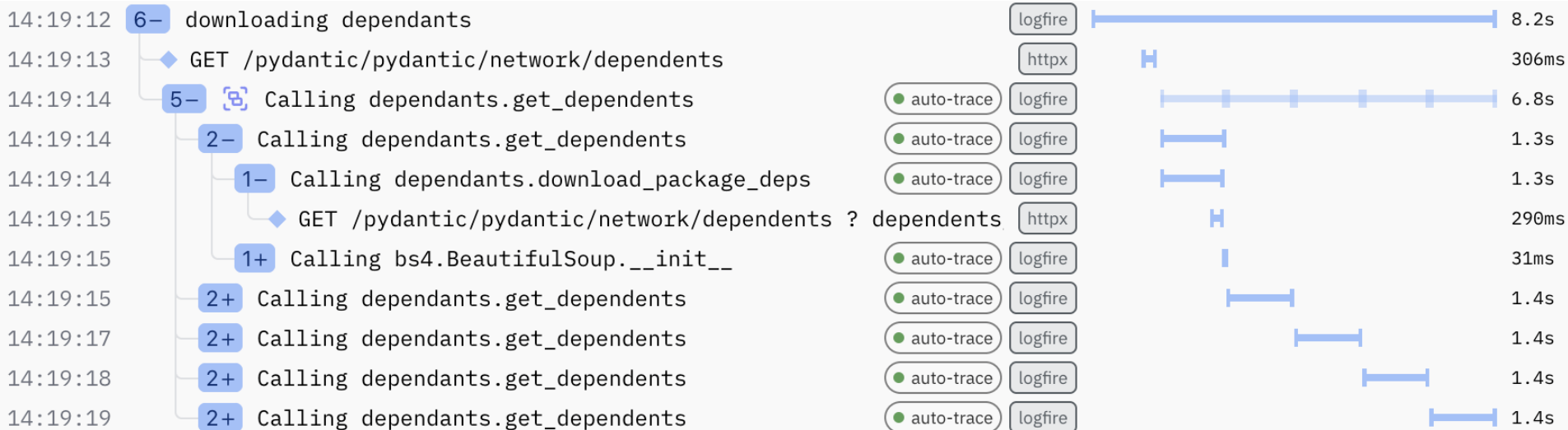
✓ {
 'fruit': 'banana',
}

Code Details

Code demo_manual.py

Auto tracing

```
logfire.install_auto_tracing(modules=[ 'dependants', 'bs4.*' ], min_duration=0.03)
```



Pretty Python objects

Pretty Python

```
1 from datetime import datetime
2 from pydantic import BaseModel
3 import logfire
4
5 logfire.configure()
6
7 class Delivery(BaseModel):
8     timestamp: datetime
9     dims: tuple[int, int]
10
11 input_json = [
12     '{"timestamp": "2020-01-02T03:04:05+00:00"}',
13     '{"timestamp": "2020-01-02T04:04:05+00:00"}',
14     '{"timestamp": "2020-01-02T05:04:05+00:00"}'
15 ]
16 deliveries = [
17     Delivery.model_validate_json(j)
18 ]
19
20 logfire.info(f'{len(deliveries)} c
```

Arguments: (as Python)

```
{
  'deliveries': [
    Delivery(
      dims=(
        10,
        20,
      ),
      timestamp='2020-01-02T03:04:05+00:00',
    ),
    Delivery(
      dims=(
        15,
        25,
      ),
      timestamp='2020-01-02T04:04:05+00:00',
    ),
    Delivery(
      dims=(
        20,
        30,
      ),
      timestamp='2020-01-02T05:04:05+00:00',
    ),
  ],
  'len(deliveries)': 3,
}
```

SQLAlchemy

Psycopg

Asyncpg

PyMongo

Sqlite3

Integration Database

`logfire.instrument_asyncpg()`

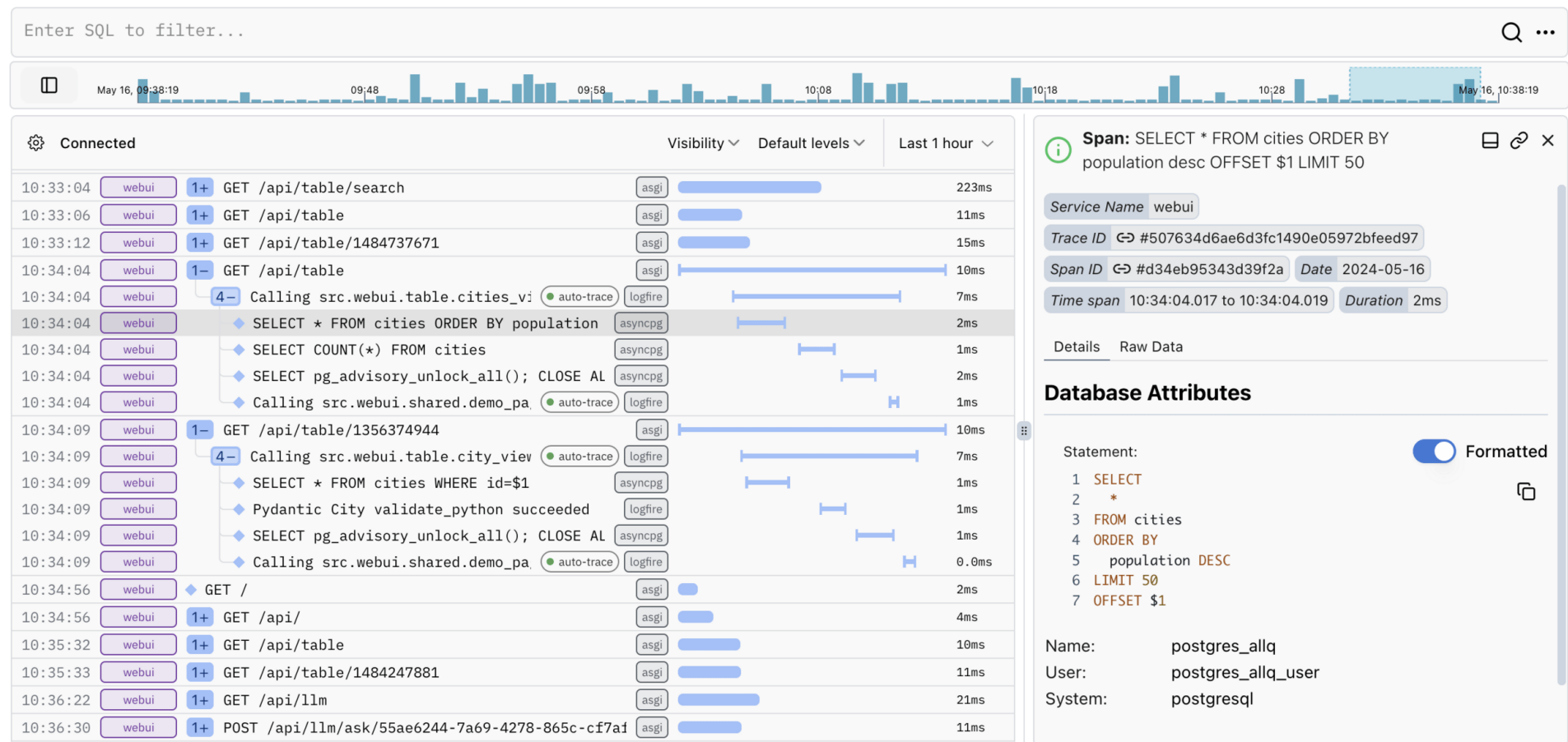
Redis

ElasticSearch

MySQL

Kafka

and more...



Integration

Database - Realworld

Error: This query is returning too many results to live stream.

Visibility ▾ Default levels ▾ Live tail ▾

10:54:51	webserver	20-	GET /admins/dash/activity/	django	1.5s
10:54:51	webserver	20-	SELECT	psycopg2	1.5s
10:54:51	webserver		SELECT "tcauth_us...approved" FROM "tcauth_user" ...[2 JOINS] WHE	psycopg2	3ms
10:54:52	webserver		SELECT "crm_role"...role_type" FROM "crm_role" WHERE ...	psycopg2	1ms
10:54:52	webserver		SELECT "crm_role"...es_person" FROM "crm_a...r" JOIN "crm_role" ON	psycopg2	1ms
10:54:52	webserver		SELECT DISTINCT "...last_name" FROM "activity_action" ...[2 JOINS]	psycopg2	1.0s
10:54:53	webserver		SELECT "activity_...t_type_id" FROM "activity_action" WHERE ... LI	psycopg2	47ms
10:54:53	webserver		SELECT "agency_no...object_id" FROM "agency_note" WHERE ... LIMIT	psycopg2	1ms
10:54:53	webserver		SELECT "activity_...on"."data" FROM "activity_action" WHERE ... LI	psycopg2	47ms
10:54:53	webserver		SELECT "activity_...t_type_id" FROM "activity_action" WHERE ... LI	psycopg2	47ms
10:54:53	webserver		SELECT "agency_no...object_id" FROM "agency_note" WHERE ... LIMIT	psycopg2	1ms
10:54:53	webserver		SELECT "activity_...on"."data" FROM "activity_action" WHERE ... LI	psycopg2	42ms
10:54:53	webserver		SELECT "activity_...t_type_id" FROM "activity_action" WHERE ... LI	psycopg2	40ms
10:54:53	webserver		SELECT "documents...items_ref" FROM "documents_document" WHERE ...	psycopg2	1ms
10:54:53	webserver		SELECT "activity_...on"."data" FROM "activity_action" WHERE ... LI	psycopg2	37ms
10:54:53	webserver		SELECT "activity_...t_type_id" FROM "activity_action" WHERE ... LI	psycopg2	36ms
10:54:53	webserver		SELECT "crm_role"...eive_smss" FROM "crm_role" WHERE ... LIMIT 21	psycopg2	1ms
10:54:53	webserver		SELECT "activity_...on"."data" FROM "activity_action" WHERE ... LI	psycopg2	35ms
10:54:53	webserver		SELECT "schedule_...s_ea_item" FROM "schedule_appointment" WHERE	psycopg2	1ms
10:54:53	webserver		SELECT "activity_...on"."data" FROM "activity_action" WHERE ... LI	psycopg2	38ms
10:54:53	webserver		SELECT "schedule_...s_ea_item" FROM "schedule_appointment" WHERE	psycopg2	1ms
10:54:53	webserver		SELECT "activity_...on"."data" FROM "activity_action" WHERE ... LI	psycopg2	35ms
10:54:52	webserver	19+	POST /api/appointments/	django	97ms
10:54:52	webserver	7+	GET /admins/dash/extra_data/	django	731ms
10:54:52	webserver		GET /static/dist/main.331ef503.js	django	3ms
10:54:52	webserver		GET /static/favicons/manifest.2e8abaff8ac.json	django	
10:54:52	webserver	10+	GET /isinfo/2659306/menu_v2.is?v=35705985c9df21349c66778631bff2bec	django	51ms

Span: SELECT "crm_role"...es_person"
FROM "crm_a...r" JOIN "crm_role" ON ...
WHERE ... LIMIT 21

Service Name webserver

Trace ID ↗ #b69fb5e0c0b2876835126bf4cab1b5ad

Span ID ↗ #5b7ee1983027d77f Date 2024-05-16

Time span 10:54:52.006 to 10:54:52.007 Duration 1ms

Details Raw Data

Database Attributes

Statement: ☒ Formatted

```
1 SELECT
2 TOP 21
3 "crm_role"."id",
4 "crm_role"."user_id",
5 "crm_role"."role_type",
6 "crm_role"."priority",
7 "crm_role"."calendar_colour",
8 "crm_role"."receive_smss",
9 "crm_administrator"."role_ptr_id",
10 "crm_administrator"."permissions",
11 "crm_administrator"."received_notifications",
12 "crm_administrator"."client_manager",
13 "crm_administrator"."default_client_view",
14 "crm_administrator"."dashboard_panels",
15 "crm_administrator"."is_support_person",
16 "crm_administrator"."is_sales_person"
17 FROM "crm_administrator"
18 INNER JOIN "crm_role"
```

django
flask
starlette
ASGI

Integration

Webframework

AWS Lambda
falcon
tornado
and more...

```
logfire.instrument_fastapi(app)
```

Connected		Visibility		Default levels		Last 15 minutes	
11:37:45	webui	1+	GET /api/table	exception	asgi		12ms
11:37:50	webui	1+	GET /api/table		asgi		10ms
11:37:52	webui	1+	GET /api/table/1156073548		asgi		12ms
11:38:21	webui	◆	GET /		asgi		2ms
11:38:21	webui	1+	GET /api/		asgi		3ms
11:38:49	webui	1+	GET /api/llm		asgi		11ms
11:38:57	webui	1+	POST /api/llm/ask/ae14f2b4-2003-43ff-ad70-8b2ac03e2291		asgi		9ms
11:38:58	webui	2+	GET /api/llm/ask/stream/ae14f2b4-2003-43ff-ad70-8b2ac03e2291		asgi		17.0s
11:39:15	webui	1+	POST /api/llm/ask/ae14f2b4-2003-43ff-ad70-8b2ac03e2291		asgi		10ms
11:39:15	webui	2+	GET /api/llm/ask/stream/ae14f2b4-2003-43ff-ad70-8b2ac03e2291		asgi		2.4s
11:39:23	webui	1+	POST /api/llm/ask/ae14f2b4-2003-43ff-ad70-8b2ac03e2291		asgi		10ms
11:39:23	webui	2+	GET /api/llm/ask/stream/ae14f2b4-2003-43ff-ad70-8b2ac03e2291		asgi		3.8s
11:39:44	webui	1+	GET /api/table		asgi		21ms
11:39:49	webui	1+	GET /api/table/search		asgi		240ms
11:39:49	webui	1+	GET /api/table		asgi		11ms
11:39:54	webui	1+	GET /api/table/1076892244		asgi		11ms
11:40:27	webui	1+	GET /api/llm		asgi		10ms
11:40:28	webui	1+	POST /api/llm/ask/747e4b48-debc-4222-949c-7d98bed08aa9		asgi		11ms
11:40:28	webui	2+	GET /api/llm/ask/stream/747e4b48-debc-4222-949c-7d98bed08aa9		asgi		13.4s
11:40:44	webui	1+	POST /api/llm/ask/747e4b48-debc-4222-949c-7d98bed08aa9		asgi		10ms
11:40:44	webui	2+	GET /api/llm/ask/stream/747e4b48-debc-4222-949c-7d98bed08aa9		asgi		12.1s
11:41:04	webui	1+	POST /api/llm/ask/747e4b48-debc-4222-949c-7d98bed08aa9		asgi		15ms
11:41:04	webui	2+	GET /api/llm/ask/stream/747e4b48-debc-4222-949c-7d98bed08aa9		asgi		23.4s
11:41:46	webui	1+	GET /api/llm		asgi		9ms
11:41:48	webui	1+	POST /api/llm/ask/69140194-fa89-42b0-8dff-31bd06386bf0		asgi		10ms
11:41:48	webui	2+	GET /api/llm/ask/stream/69140194-fa89-42b0-8dff-31bd06386bf0		asgi		13.3s
11:42:05	webui	1+	POST /api/llm/ask/69140194-fa89-42b0-8dff-31bd06386bf0		asgi		10ms
11:42:05	webui	2+	GET /api/llm/ask/stream/69140194-fa89-42b0-8dff-31bd06386bf0		asgi		27.4s

Span: GET /api/table

DetailsRaw Data

HTTP Request Attributes

GEThttp://10.214.84.6:8000/api/table → 200

Request Headers: 18 header(s) ^

accept: */*
accept_encoding: gzip
cdn_loop: cloudflare; loops=1; subreqs=1
cf_connecting_ip: 54.191.253.12
cf_connecting_o2o: 1
cf_ew_via: 15
cf_ipcountry: US
cf_ray: 884c76550692c3e9-SEA
cf_visitor: {"scheme":"https"}
cf_worker: onrender.com
host: demo.logfire.dev
render_proxy_ttl: 4
rndr_id: 8d19a563-65c8-43de
true_client_ip: 54.191.253.12
user_agent: python-httpx/0.27.0
x_forwarded_for: 54.191.253.12,54.191.253.12, 172.71.151.62
x_forwarded_proto: https
x_request_start: 1715873984815742

Response Headers: 2 header(s) ^

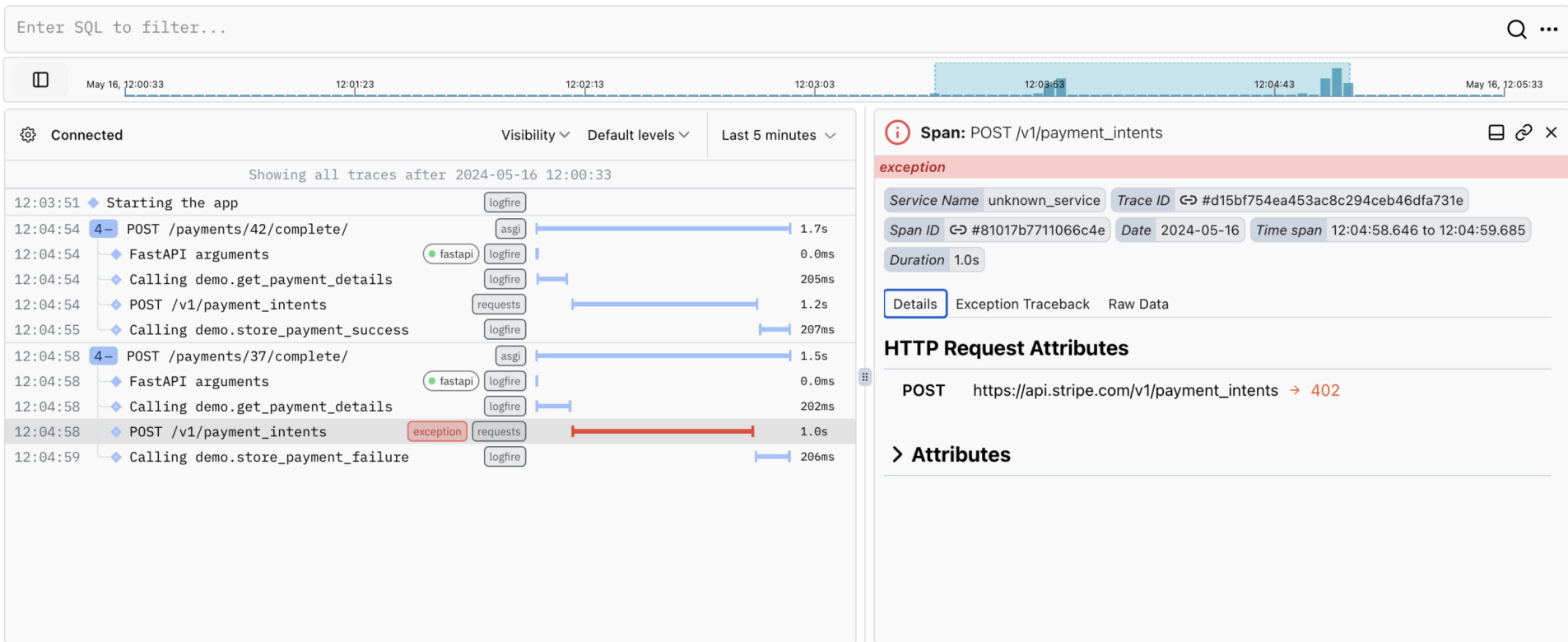
content_length: 11309
content_type: application/json

> Attributes

httpx
requests
aiohttp

Integration

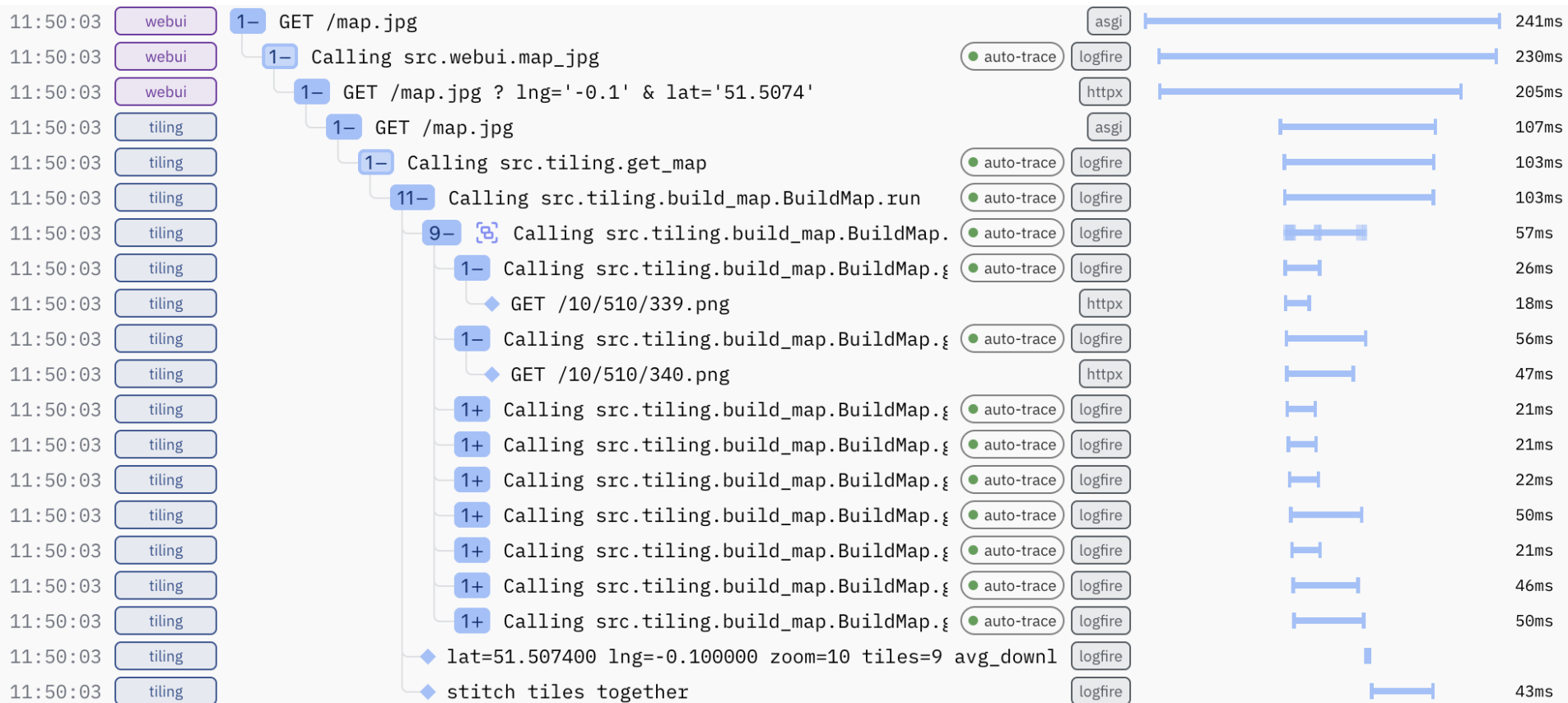
HTTP Requests



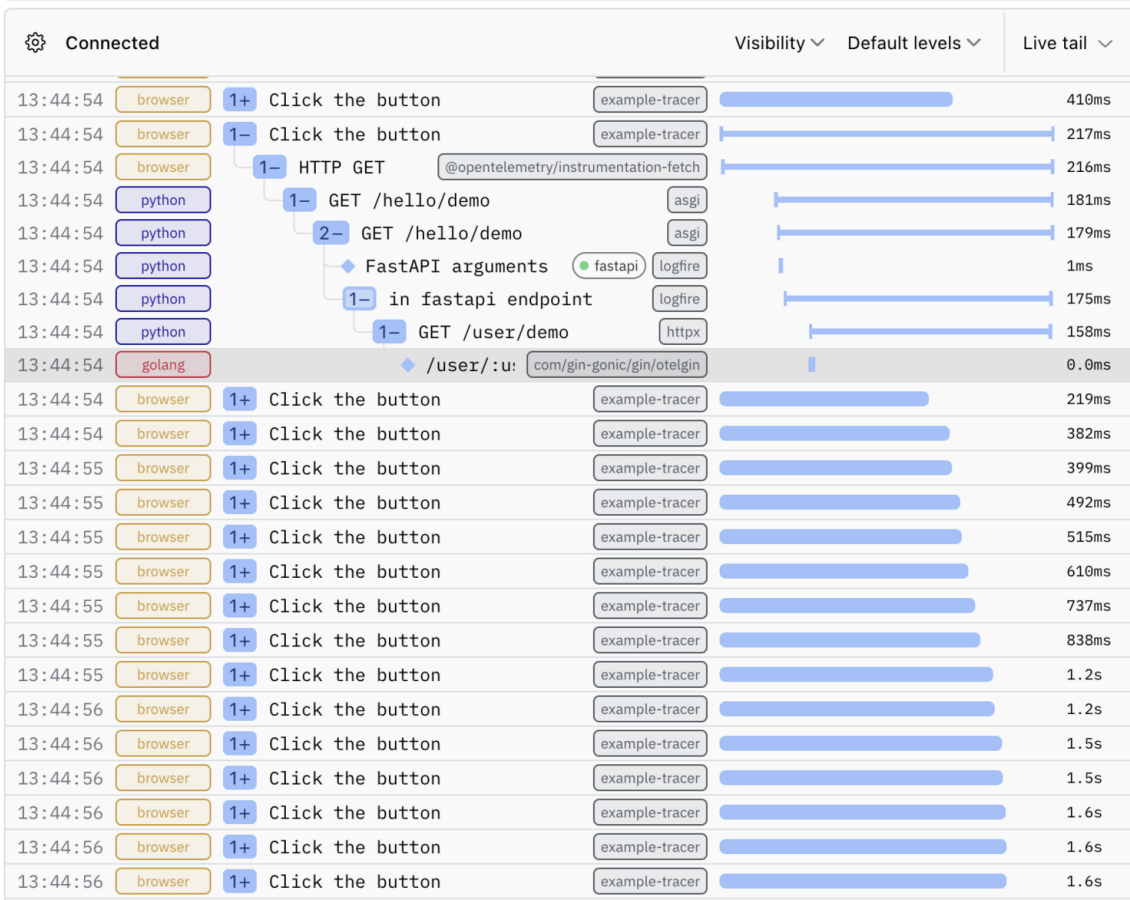
```
17         store_payment_failure(user_id)
18         return JsonResponse(content={'detail': 'Card error'},
19                               status_code=400)
20     else:
21         store_payment_success(user_id, intent)
```

OpenTelemetry win

Distributed Tracing



OpenTelemetry win Other Languages



```
21    });  
22    });  
23    };
```

Span: /user/:user

Service Name: goLang Trace ID: #f16f585cdee21863c51e027d4eac7c43

Span ID: #4e9b3339035a5cdf Date: 2024-05-16

Time span: 13:44:54.569 to 13:44:54.569

Details Raw Data

HTTP Request Attributes

GET → 200

User Agent: python-httpx/0.27.0

Attributes

11 items

```
{  
  "http.route": "/user/:user"  
  "http.method": "GET"  
  "http.scheme": "http"  
  "http.target": "/user/demo"  
  "net.host.name": "otel-otlp-go-service"  
  "net.host.port": 8080  
  "http.status_code": 200  
  "net.sock.peer.addr": "::1"  
  "net.sock.peer.port": 60354  
  "user_agent.original": "python-httpx/0.27.0"  
  "net.protocol.version": "1.1"  
}
```

Integration

Pydantic

12:27:09	◆ Pydantic Delivery validate_json succeeded	logfire	1ms
12:27:09	◆ Pydantic Delivery validate_json succeeded	logfire	1ms
12:27:09	◆ Pydantic Delivery validate_json succeeded	logfire	1ms
12:27:09	◆ Pydantic Delivery validate_json failed	logfire	0.0ms

Span: Pydantic Delivery validate_json failed

Details Raw Data

Arguments: (as Python)

```
{ 6 items
  errors: [
    { 4 items
      loc: (
        dims,
        1,
      ),
      msg: 'Field required',
      type: 'missing',
      input: [
        '10',
      ],
    },
  ],
  success: False,
  input_data: '{"timestamp": "2020-01-02T03:04:05Z", "dims": ["10"]}',
  error_count: 1,
  schema_name: 'Delivery',
  validation_method: 'validate_json',
```

rd='all'))

}',
'}',
'}',
'}',

```
1 from dataclasses import dataclass
2 from pydantic import BaseModel
3 import logfire
4
5 logfire.configure()
6
7 class Delivery(BaseModel):
8     timestamp: str
9     dims: List[str]
10
11 input_data = {
12     "timestamp": "2020-01-02T03:04:05Z",
13     "dims": ["10"],
14 }
15
16 delivery = Delivery(input_data)
17
18 # Delivery validation failed
19 delivery.validate_json()
20
21 )
```

Integration OpenAI

Showing all traces after 2024-05-16 12:38:38

12:40:28	2-	Picture of a cat in the style of a famous painter	exception	logfire	12.0s
12:40:28		Chat Completion with 'gpt-4'		openai	5.9s
12:40:34	◆	Image Generation with 'dall-e-3'	exception	openai	6.1s
12:41:52	2+	Picture of a cat in the style of a famous painter		logfire	16.7s
12:42:28	2-	Picture of a cat in the style of a famous painter		logfire	12.7s
12:42:28	◆	Chat Completion with 'gpt-4'		openai	1.1s

Span: Image Generation with 'dall-e-3'

exception

Service Nameunknown_serviceTrace ID↔ #f8ca09776f61ed38733d828bcea4bf5bSpan ID↔ #920d231e8f8563b8Date2024-05-16

Time span12:40:34.640 to 12:40:40.705Duration6.1s

DetailsException TracebackRaw Data

BadRequestError

Error code: 400 - {'error': {'code': 'content_policy_violation', 'message': 'Your request was rejected as a result of our safety system. Image descriptions generated from your prompt may contain text that is not allowed by our safety system. If you believe this was done in error, your request may succeed if retried, or by adjusting your prompt.', 'param': None, 'type': 'invalid_request_error'}}

Exception stack trace:

Traceback (most recent call last):
File "/Users/samuel/code/logfire-demo/.venv/lib/python3.12/site-packages/logfire/_internal/integrations/openai.py", line 124, in instrumented_openai_request
response = on_response(original_request_method(**kwargs), span)
File "/Users/samuel/code/logfire-demo/.venv/lib/python3.12/site-packages/openai/_base_client.py", line 988, in _request
raise self._make_status_error_from_response(err.response) from None
openai.BadRequestError: Error code: 400 - {'error': {'code': 'content_policy_violation', 'message': 'Your request was rejected as a result of our safety system. Image descriptions generated from your prompt may contain text that is not allowed by our safety system. If you believe this was done in error, your request may succeed if retried, or by adjusting your prompt.', 'param': None, 'type': 'invalid_request_error'}}

Customization

Custom UI

Record data

Reset

```
1 {
2   "attributes": {
3     "async": false,
4     "request_data": {
5       "model": "gpt-4",
6       "stream": true,
7       "messages": [
8         {
9           "role": "system",
10          "content": "Response entirely in plain text"
11        },
12        {
13          "role": "user",
14          "content": "please write a fun fantastical scenario in around 100 words"
15        }
16      ]
17    }
18  }
```

Format

Custom panel function

Reset

```
1 import { z } from 'zod'
2
3 import { CustomPanel, MarkdownText, Section, Image, JsonObject } from './Custom
4 type Record = any
5
6 const toolCallSchema = z.object({
7   function: z.object({
8     arguments: z.string(),
9     name: z.string(),
10   }),
11 })
12
13 const messageSchema = z.object({
14   role: z.string(),
15   content: z.string().nullable()
```

Render

Rendered panels

☐ Show type declarations

LLM Chat Completions

gpt-4

system

Response entirely in plain text

user

please write a fun fantastical scenario in around 100 words

assistant

Tool call: GetCurrentWeather

```
{ 1 items
  "location": "Tokyo"
}
```

assistant

In the heart of an enchanted forest twinkling with pixie dust, a village of mushroom houses existed. The occupants were tiny fairy folk that rode on dragonflies, each with a thimbleful of magical abilities. One day, they hosted a Great Bake-Off, but the baker, Mr. Finch, couldn't find his magic spatula. Suddenly, a baby unicorn trotted into the village square carrying the lost spatula, a cheeky grin on her face. The fairies cheered and the biggest tart cherry pie was baked and shared, and the unicorn got the first slice for her cheeky service.

LlmImagesPanel would not be rendered for this record

Explore SQL

Tab

+

May 16, 12:59

×

```
select start_timestamp, (attributes->'response_data'->'usage'->'total_tokens')::int as usage, attributes->'response_data'->'message'->'content' as message
from records
where otel_scope_name='logfire.openai' and attributes->'response_data'->'message' ? 'content'
order by start_timestamp desc
```

	start_timestamp	usage	message
1	2024-05-16T16:42:28.324879Z	36	Pablo Picasso
2	2024-05-16T16:41:52.587113Z	39	Giotto di Bondone
3	2024-05-16T16:40:28.731531Z	99	There were several influential painters in the 20th century, but one of the most recognized is Pablo Picasso. Known for his role in establishing t

Returned 3 rows in 32 milliseconds.

Limit: 100 Time window: Last 1d

May 16, 12:59

×

```
select sum((attributes->'response_data'->'usage'->'total_tokens')::int) from records
where otel_scope_name='logfire.openai'
```

	sum
1	174

Returned 1 row in 44 milliseconds.

Limit: 100 Time window: Last 1d



**We're here all week,
to help you with
Pydantic Logfire**

And on slack after that
pydantic.dev/logfire

Thank you