

EPITA · 2026

Observability 101

Session 1: Why observability?



DATADOG

What do you think observability is?

01

What is observability?



02

How is it different from “monitoring”?

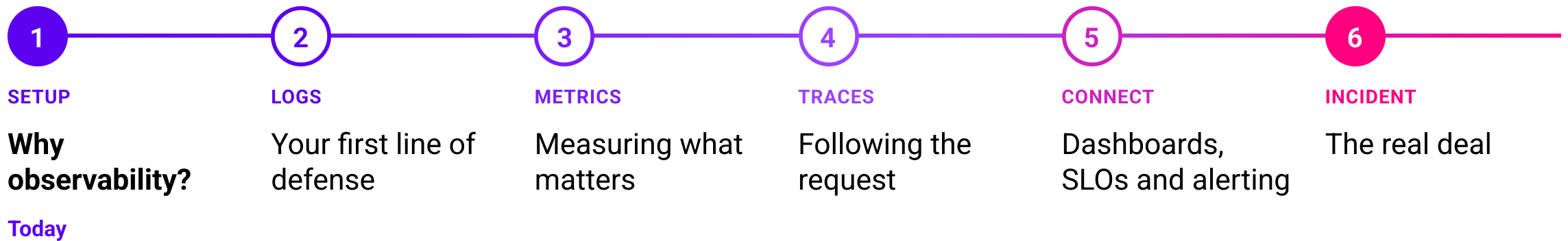


03

What is it for? Who uses it, and when?



The course in one picture



Your instructors



Elliott

Software Engineer @ Datadog

Open-source Go instrumentation. Securing customers applications and Kubernetes clusters.

EPITA PROMO

2023



Thomas

Software Engineer @ Datadog

Infrastructure. Owns all of Datadog's Envoy proxies, including the ones ingesting all customer traffic: logs, metrics, traces.

EPITA PROMO

2024

Why Datadog?

It is where we work and what we know best. And each group gets a full Datadog organization, for free.

The instrumentation you write uses OpenTelemetry, which is vendor-neutral.

The concepts are the point. The tool is interchangeable.

	Open source	Datadog
Instrumentation	OpenTelemetry	OpenTelemetry
Logs	Loki or OpenSearch	Log Management
Metrics	Prometheus	Metrics
Traces	Jaeger or Tempo	APM
Dashboards	Grafana	Dashboards

Leave the AI assistant closed in this course.

The goal is to learn to investigate a system, not to watch a tool do it.

Debugging intuition comes from digging yourself: a hypothesis, being wrong, finding the evidence.

During an incident, the person who understands the system is the one who fixes it.

What do you think observability is?

01

What is observability?



02

How is it different from “monitoring”?



03

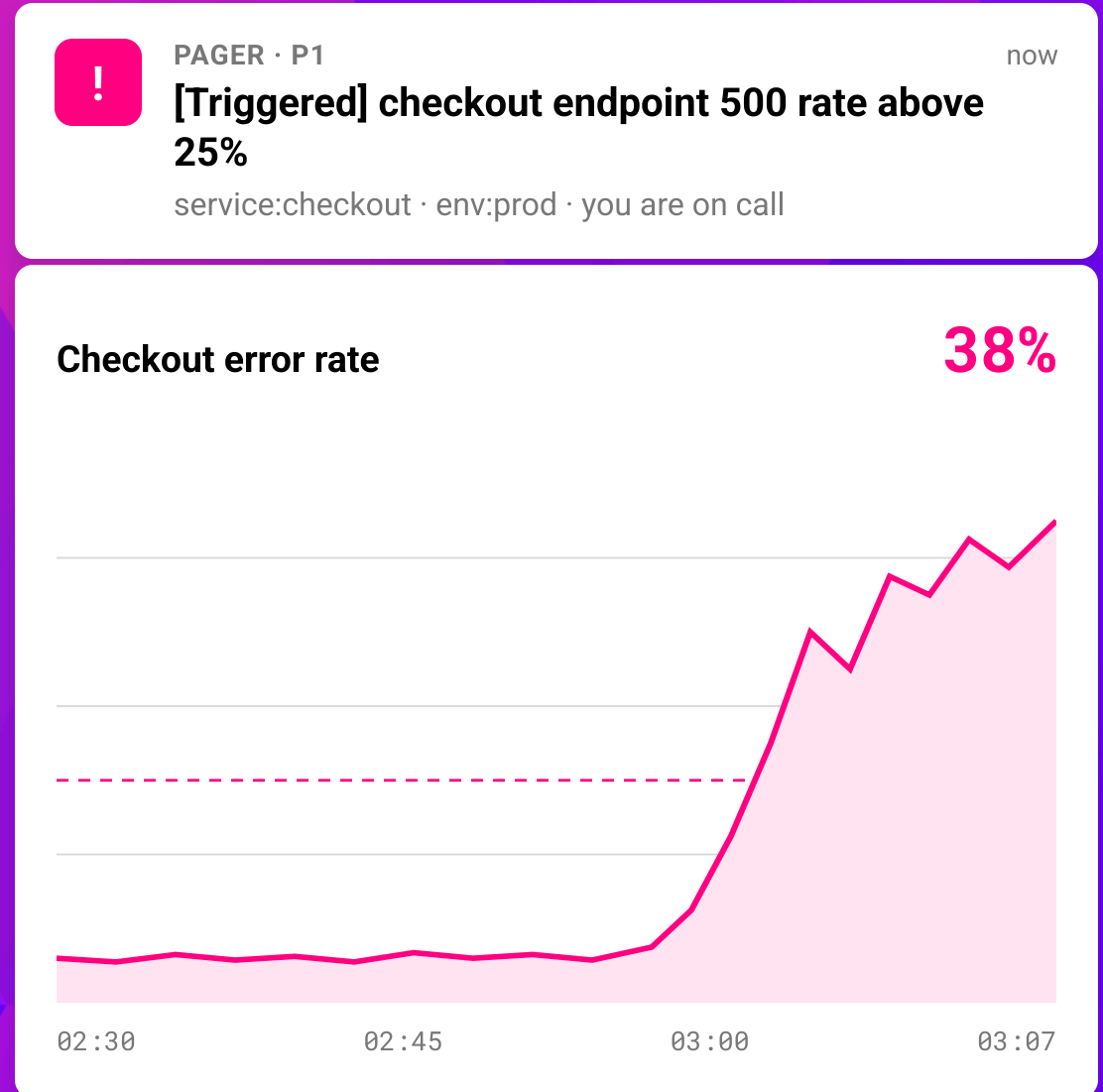
What is it for? Who uses it, and when?



03:07

It's 3 a.m. Checkout is failing.

Where do you look?



Monitoring vs observability

Monitoring

Answers questions you thought of in advance. Known unknowns.

"Is the CPU above 90%?"

"Is the error rate above 5%?"

Observability

Lets you ask new questions from the outside, without shipping new code. Unknown unknowns.

"Why are only big baskets failing, and only for some users?"

Monitoring tells you **that** something is wrong. Observability helps you find out **why**.

Why it matters now: one request, many services

In a monolith, a request lives in one process: a stack trace or a log file is enough. In a distributed system, it crosses many.

ONE PAID CHECKOUT ON YOUR STACK

67 calls in
a single trace

7
services

8
Postgres queries

3
Redis calls

POST /checkout · trace waterfall



The three pillars, at a glance

Logs

What happened, event by event.

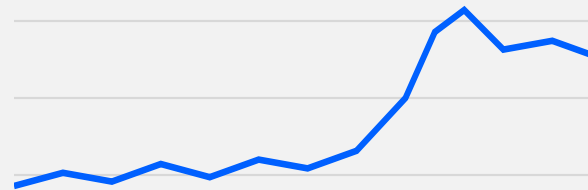
```
{
  "timestamp": "03:07:12.481Z",
  "level": "error",
  "service": "billing",
  "message": "payment timeout"
}
```

Session 2

Metrics

How much and how often, over time.

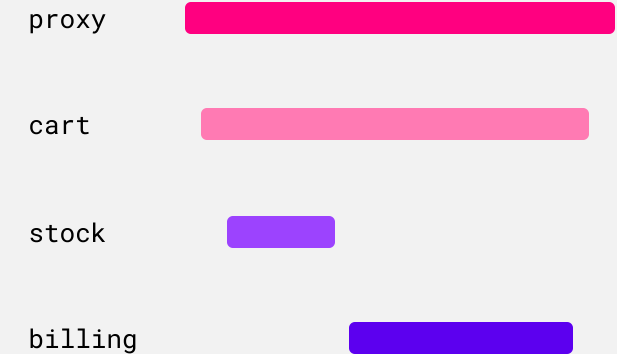
p95 latency, ms



Session 3

Traces

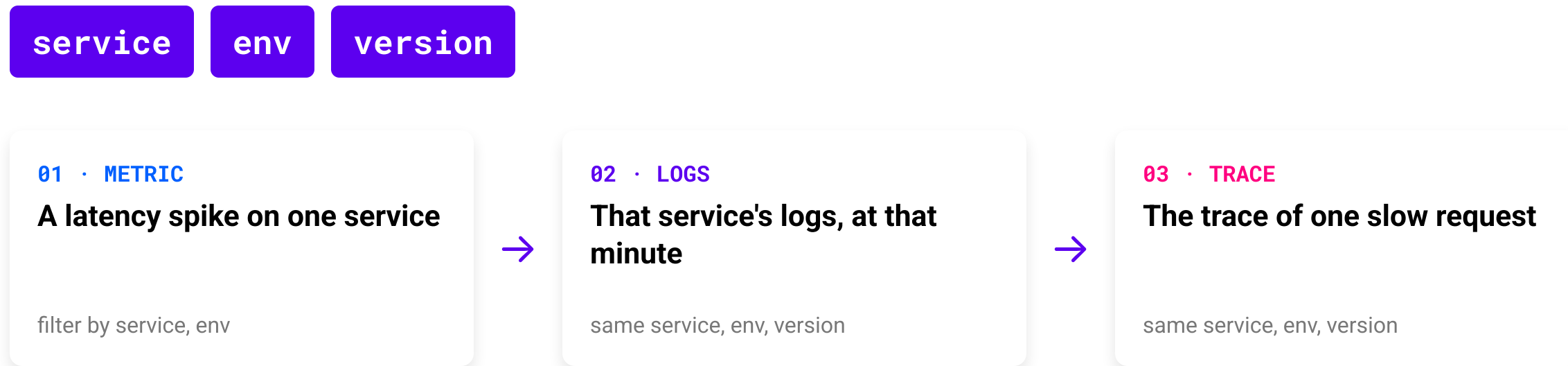
Where the time went, across services, for one request.



Session 4

How the pillars connect

On their own, they are three disconnected views. Shared tags tie them together: the unified service tags.



The tool never checks your tags. A wrong tag is not an error: the data just lands in the wrong place.

Questions?

Monitoring vs observability, microservices, the three pillars and how they connect.

Knight Capital

\$440M

lost in roughly 45 minutes

A deployment missed one of eight servers, which ran old, dormant trading code. Automated emails flagging an error went out before the market opened. Nobody acted on them.

After an incident, observability is how you write the post-mortem and size the damage.

Cloudflare

27 MIN

of errors on Cloudflare-proxied sites, worldwide

A single WAF rule with a badly backtracking regular expression was deployed globally. CPU hit 100% on every machine serving HTTP traffic.

In a real incident you get one shot. Every wrong guess costs time and money.

AWS S3, us-east-1

4 HOURS

of S3 disruption in the region, taking a large part of the web with it

An engineer mistyped an input while removing a few servers, and removed far more. AWS's own status dashboard depended on S3, so it could not show the outage.

Set up observability before the incident. Without it, you are blind when it matters.

Before, during and after an incident, the question is not “is it broken?” but “why, and where?”

What observability is for, beyond outages

- | | | |
|----|--|---|
| 01 | Find the root cause faster | Lower your mean time to resolution (MTTR). |
| 02 | Know the blast radius | Who is affected, and how badly? |
| 03 | Confirm a fix worked | Evidence, not hope. |
| 04 | Catch problems early | Before your users do. |
| 05 | Understand a system you did not write | That is exactly your situation, from today. |

So, what is observability?

01

What is observability?



Today: asking new questions about your system from the outside, without shipping new code.

02

How is it different from “monitoring”?



Today: monitoring tells you that something is wrong. Observability helps you find out why.

03

What is it for? Who uses it, and when?



Today: Everybody having a question on their system. From engineers running incidents to CEOs marketing KPIs

Questions?

Anything from the war stories, or about observability in general.

The lab starts here

The stack you inherit

Your group inherits a small marketplace already in production, with customers placing orders every minute.

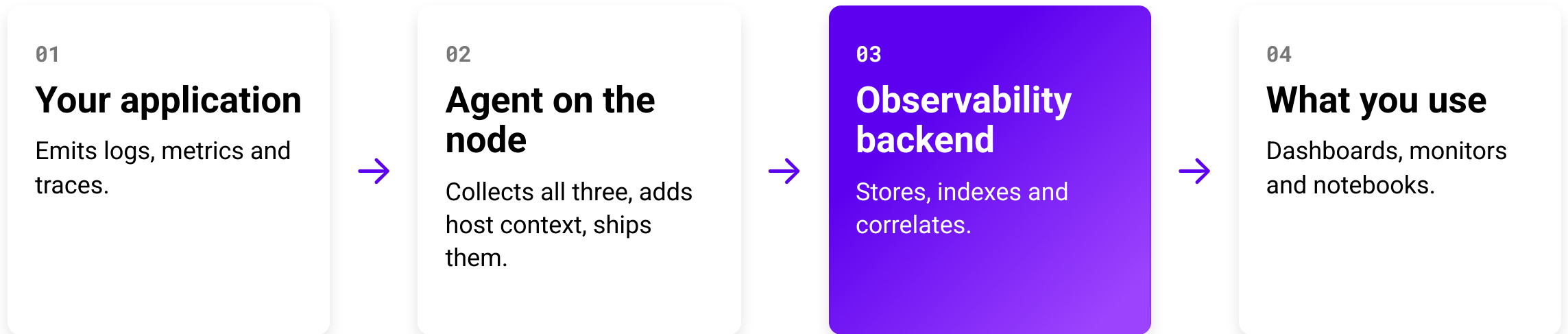
The previous team left almost no documentation and almost no instrumentation.

Over six sessions, you make it observable.



Things will break along the way. Investigating what broke is part of the job.

How data gets there



How it works and how you are graded

- **Groups of 4.** Each group has its own deployed copy of the stack and its own Datadog organization.
- **Theory and hands-on, every session,** on your own stack.
- **Work carries over.** What you start in a session, you finish with your group before the next one.
- **Every investigation is a Datadog Notebook** in six sections. Evidence and verification are live queries, never screenshots.

SYMPTOM

HYPOTHESIS

EVIDENCE

ACTION

VERIFICATION

CONCLUSION

- **Every fix goes through a merge request.**

Grading

All group work

Final incident response

Session 6

50%

Incident investigations

Sessions 2 to 5

30%

Instrumentation quality

Sessions 2 to 5

20%

Get your bearings

01

Sign in

To your team's Datadog organization.

02

Meet your marketplace

The services you just inherited.

03

See what already flows in

And what is missing.

04

Question the data

How far can you trust what the tool tells you?

HAND-IN

Before session 2

A merged merge request

A Datadog Notebook named “**Session 1: inventory**”

Questions?

The stack, the notebooks, the grading, or today's lab. We are around during the whole session.