

Backstage | CON
NORTH AMERICA

Instrumenting Backstage with OpenTelemetry: From Zero Visibility to Full Traces in Minutes

Mithun Banerjee | *Agentic Solution Architect, Splunk (Cisco)*

Meet Maya

1

Platform engineer.

She owns and operates Backstage for her company.

2

Everything is healthy.

Catalog in sync, templates running, plugins integrated.

3

Developers rely on it.

Every new service starts from her golden paths.

The Week Backstage Slowed Down

1

Monday: templates stall.

New services take minutes, then never finish.

2

Wednesday: registrations fail.

Catalog timeouts. The help channel floods.

3

Thursday: the escalation.

“Should we replace Backstage?”

What Was at Stake

This isn’t a slow page.
Trust is on the line.

- Teams bypass golden paths
- Support queue overflows
- Leadership doubts the platform

Four days.
Every dashboard green.
No idea why.

Monitoring tells you it's broken.

Observability tells you why.

Monitoring
Known failure modes, dashboards, alerts

Observability
New questions, no new code

Built from telemetry
Metrics, traces and logs

Three Signals

Metrics

What is wrong

e.g. error rate or latency is spiking

Traces

Where it's wrong

e.g. which call in the request is slow

Logs

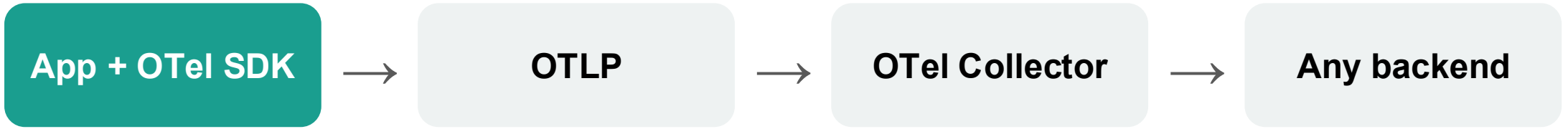
Why it's wrong

e.g. the actual error message

The value is the correlation.

Instrument once.
Send it anywhere.

Open standard · CNCF project · vendor-neutral



The One File

One file. Zero changes to Backstage core.

```
const sdk = new NodeSDK({
  resource: resourceFromAttributes({
    [ATTR_SERVICE_NAME]: 'backstage-backend',
  }),
  traceExporter: new OTLPTraceExporter(),
  metricReaders: [
    new PeriodicExportingMetricReader({
      exporter: new OTLPMetricExporter(),
    }),
  ],
  instrumentations: [getNodeAutoInstrumentations()],
});

sdk.start();
```

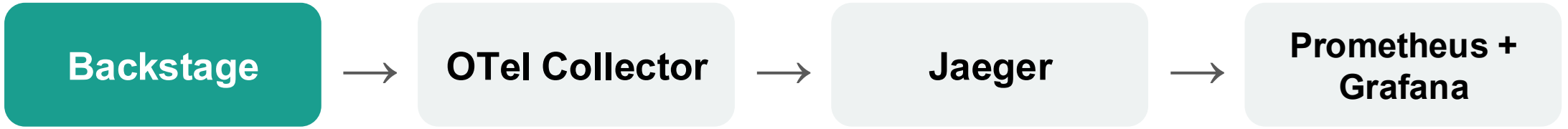
Excerpt of app/packages/backend/src/instrumentation.ts

Rebuilding the Incident

Backstage calls a real dependency



Backstage sends telemetry out

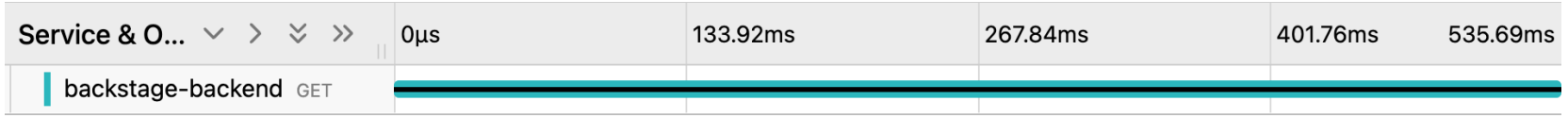


Healthy Baseline: ~0.5 s

One click. Two external calls. Both traced.

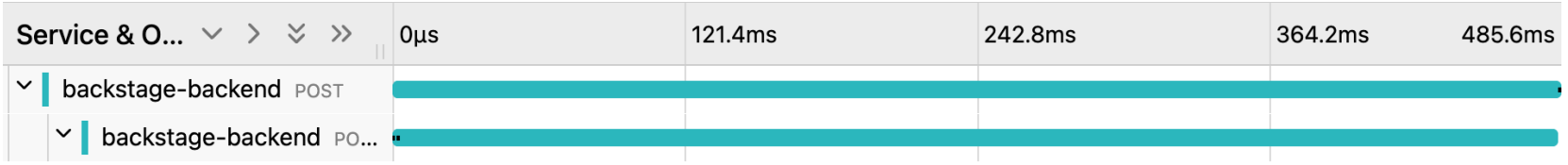
536 ms

Git fetch · 1 span



486 ms

Catalog register · 76 spans

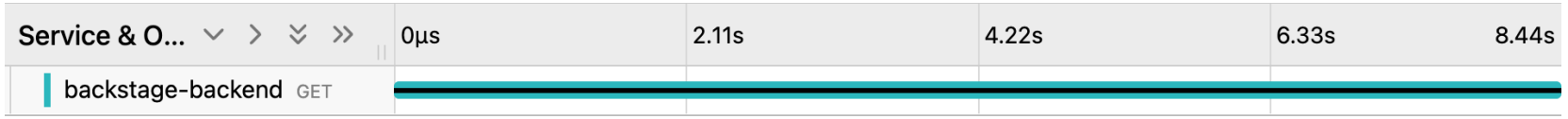


The Incident: 16x Longer

Same click. Same traces. 16x longer.

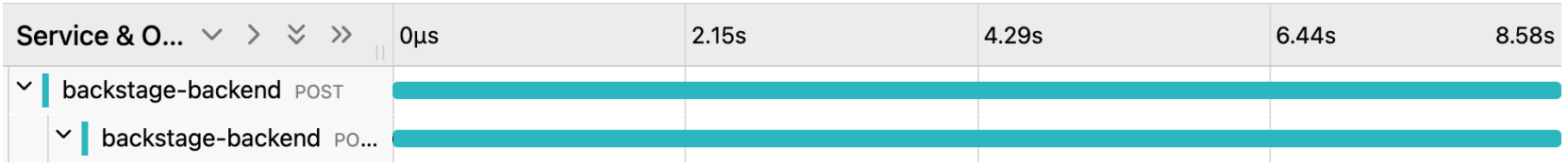
8.44 s

Git fetch · was 0.54 s (~16x)



8.58 s

Catalog register · was 0.49 s (~18x)



It wasn't Backstage.
It was Gitea.
+8 s on every Git call.

The Outcome

Four days of guessing.
One trace to the answer.

Evidence to the Gitea owners

Fixed in hours, not days

Backstage keeps its trust

Key Takeaways

1. One file. Full visibility.
2. The trace points at the culprit.
3. Evidence ends the blame game.

Thank You

Backstage CON NORTH AMERICA

Questions?

Repo: github.com/mithun100/backstage-otel-kubecon-demo

Connect: linkedin.com/in/mbanerjee

Scan to give feedback



Instrument Backstage with OpenTelemetry: From Zero Visibility to Full Traces in Minutes