AUTOMATING INSIGHTS:

BUILDING SCALABLE OBSERVABILITY WITH OPENTELEMETRY AND HASHICORP

HASHICORP USER GROUP BENGALURU MEETUP
JAN 31st 2026

new relic

👋 **I'm Jones Zachariah Noel N (zachjonesnoel)**

- 🥑 Senior DevRel Engg @ New Relic
- ☁️ AWS Serverless Hero
- ⚡ Serverless architect
- 🚀 AWS UG Bengaluru co-organizer
- 👨‍💻 Runs newsletter / blog on The Serverless Terminal
- ▶ Co-run The Zacs' Show Talking AWS podcast

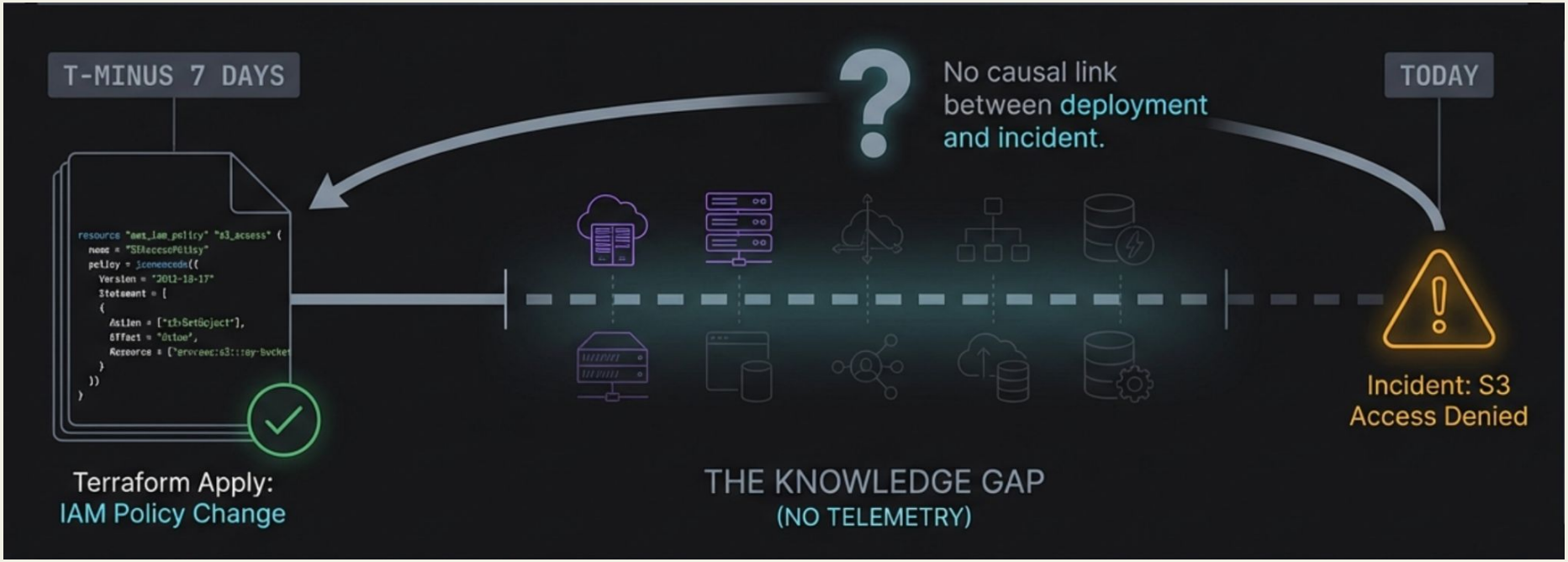
JONES ZACHARIAH NOEL N
Senior Developer Relations Engineer, APJ
New Relic

01 OpenTelemetry 101

02 Monitoring with Hashicorp in scale

03 Best practices

You think you don't need observability until.....

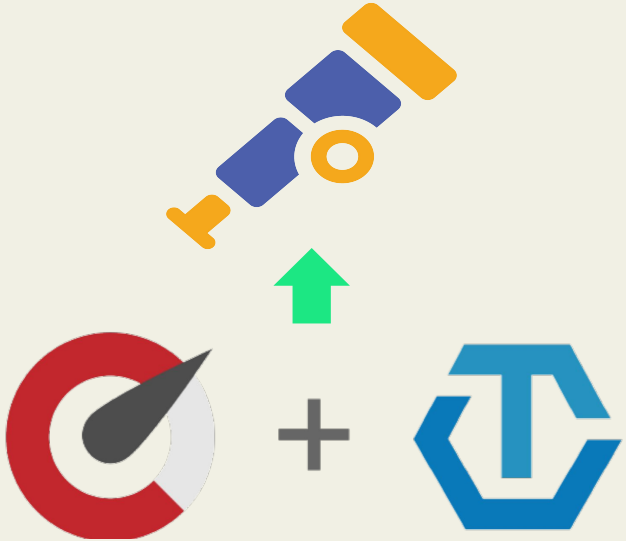


Generated with Nano Banana

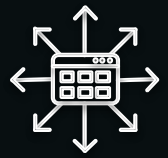
ith Nano Banana

Key Facts on OpenTelemetry

- OpenTelemetry is an Incubating project of CNCF.
- Formed through a merger of the OpenTracing and OpenCensus projects.
- Vendor agnostic - set of APIs, libraries, integrations, and a collector service for telemetry.
- Standardizes how you collect telemetry data from your applications and services.
- Send it to an Observability platform of your choice.



The Rise of OpenTelemetry



Ubiquity

Promotes better coverage for instrumentation

Vendor Neutral

Provides flexibility to change backend

Interoperable

End-to-end visibility with standard instrumentation

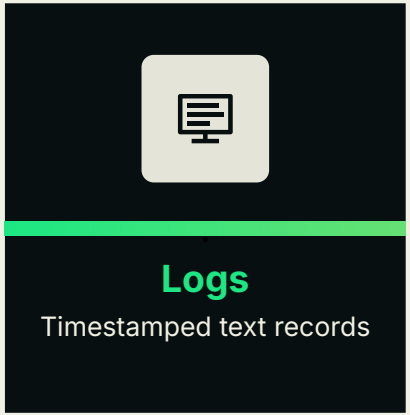
Configurable

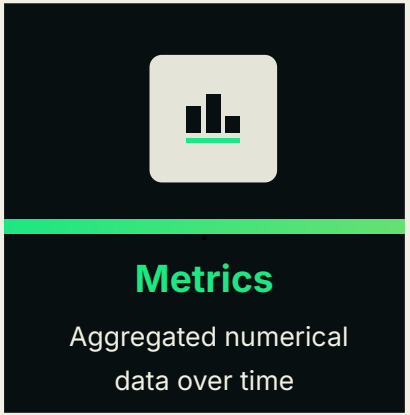
Pick and choose from the pieces what is needed

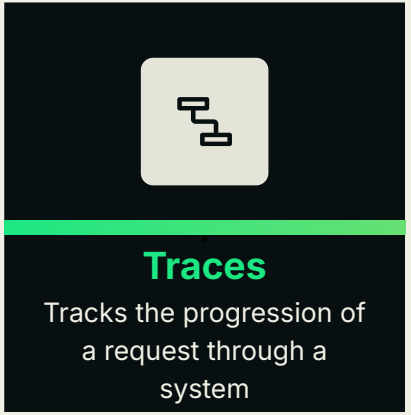
By 2025, **70%** of new cloud-native application monitoring will use open source instrumentation

[1] Source: Gartner Magic Quadrant 2021 for Application Performance Monitoring - **link**

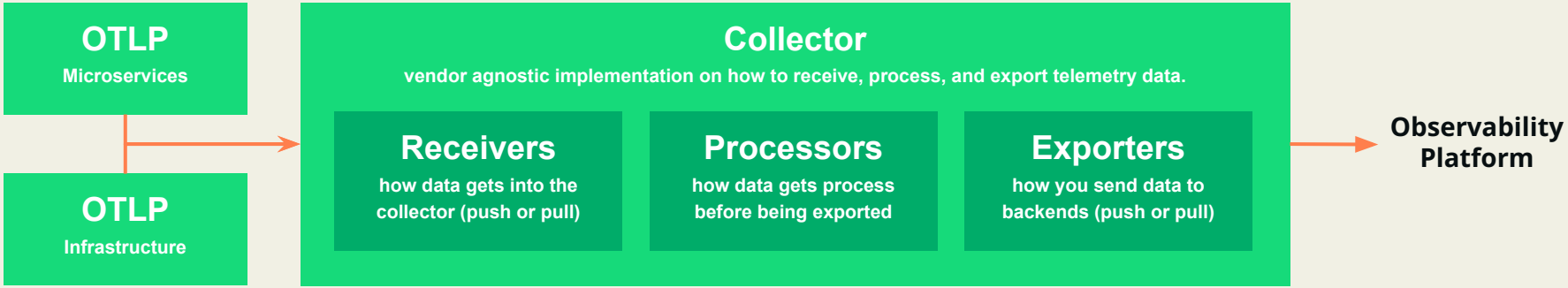
OpenTelemetry 101



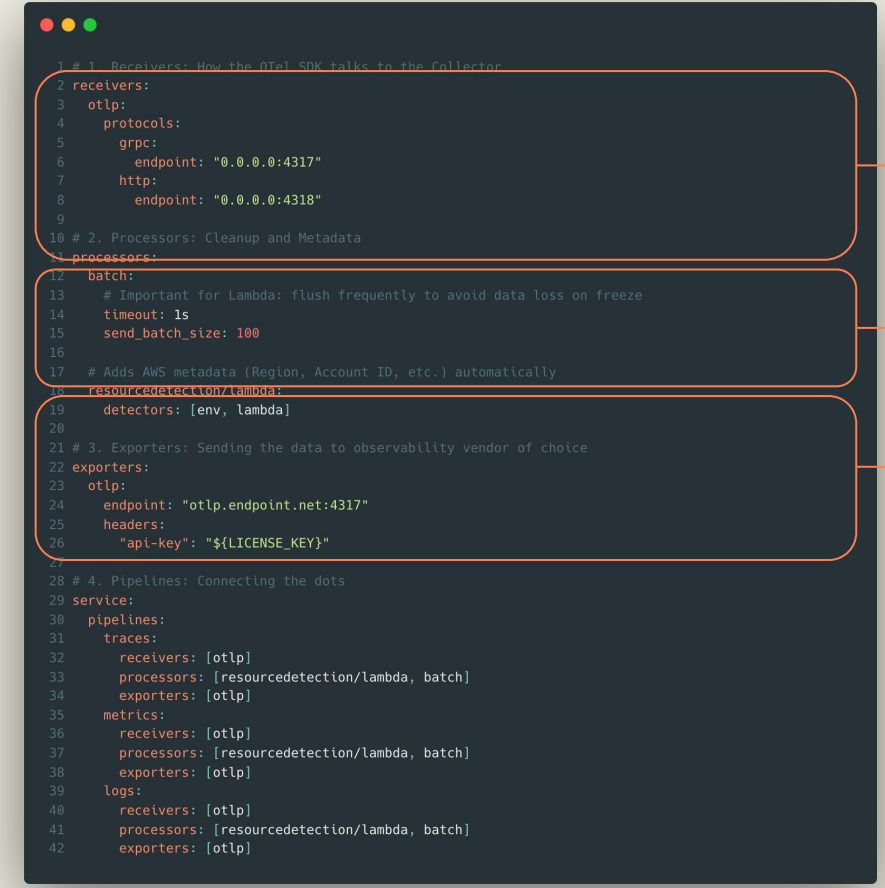




Collector configuration



a **proxy** that process multiple telemetry formats, via an **agent** or a **gateway**
(e.g., OTLP, Jaeger, Prometheus, as well as many commercial/proprietary tools)



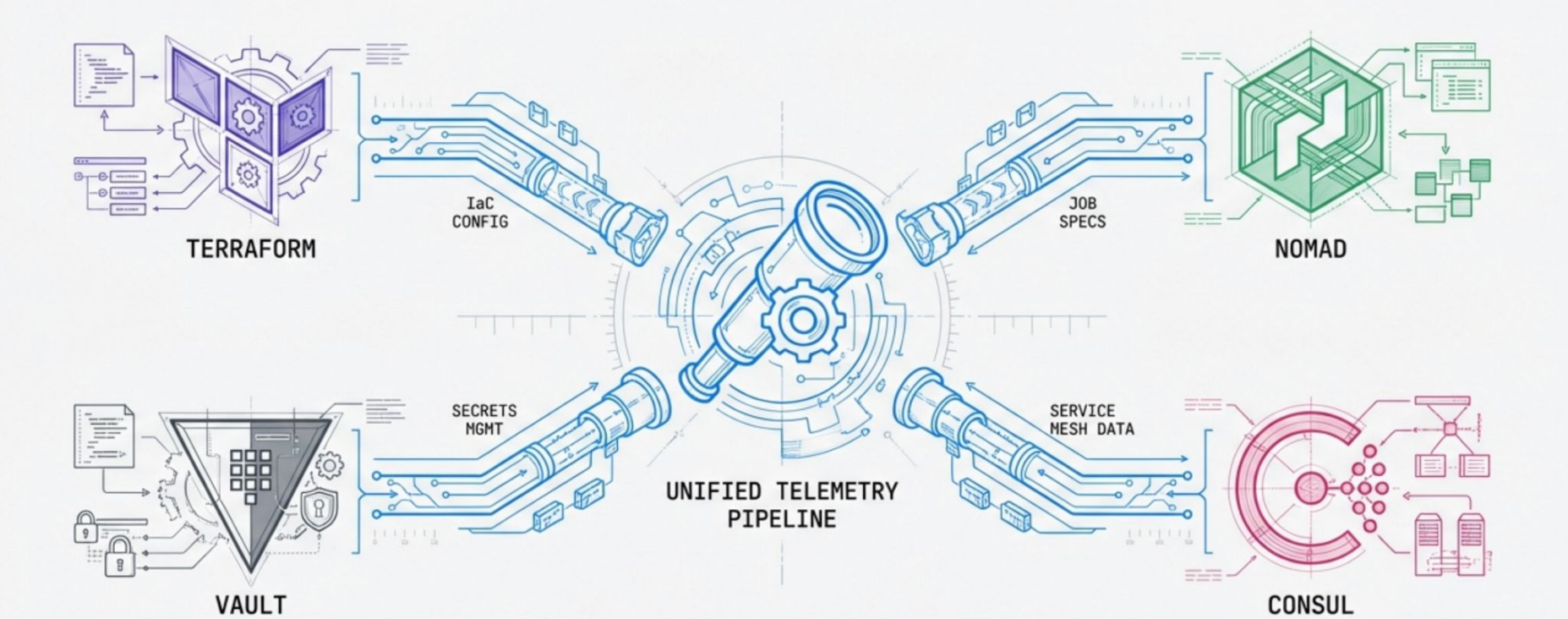
```yaml
# 1. Receivers: How the OTel SDK talks to the Collector
receivers:
  otlp:
    protocols:
      grpc:
        endpoint: "0.0.0.0:4317"
      http:
        endpoint: "0.0.0.0:4318"

# 2. Processors: Cleanup and Metadata
processors:
  batch:
    # Important for Lambda: flush frequently to avoid data loss on freeze
    timeout: 1s
    send_batch_size: 100

  # Adds AWS metadata (Region, Account ID, etc.) automatically
  resourcedetection/lambda:
    detectors: [env, lambda]

# 3. Exporters: Sending the data to observability vendor of choice
exporters:
  otlp:
    endpoint: "otlp.endpoint.net:4317"
    headers:
      "api-key": "${LICENSE_KEY}"

# 4. Pipelines: Connecting the dots
service:
  pipelines:
    traces:
      receivers: [otlp]
      processors: [resourcedetection/lambda, batch]
      exporters: [otlp]
    metrics:
      receivers: [otlp]
      processors: [resourcedetection/lambda, batch]
      exporters: [otlp]
    logs:
      receivers: [otlp]
      processors: [resourcedetection/lambda, batch]
      exporters: [otlp]
```

Instrumented my application ✅

GIF from Tenor

Infrastructure?

Unified Observability for your Hashicorp ecosystem



Generated with Nano Banana

Observability pillars in context of infrastructure

Logs

Logs are no longer flat text files; they are structured JSON events containing Trace IDs. This allows you to filter logs by a specific deployment ID or developer commit.

Metrics

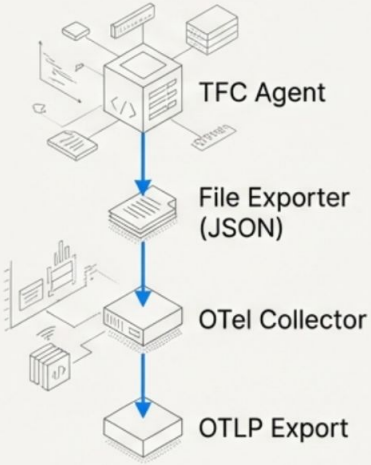
Compare the "Declared State" (Terraform code) vs. "Actual State" (Cloud reality). If Terraform thinks there are 3 nodes but CloudWatch reports 4, OTel metrics expose this discrepancy

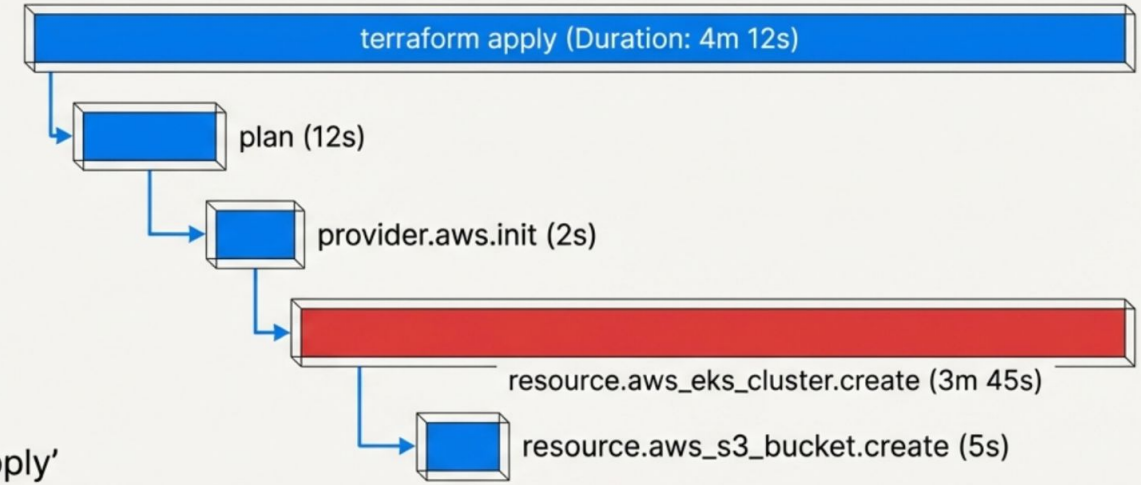
Traces

Visualize a terraform apply as a parent span. Child spans represent individual resource creations. If a K8s cluster takes 9 minutes to provision, it appears as a distinct long bar in the trace view

Tracing your terraform apply

Treating Infrastructure Deployment as a Software Workflow

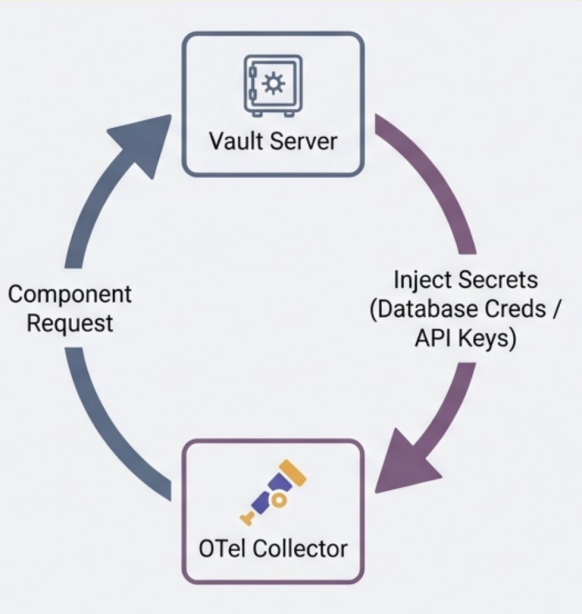




Insight: Every ‘plan’ and ‘apply’ becomes a traceable span. Correlate long infrastructure durations with downstream application latency.

Best Practices

Using Vault to retrieve the secrets




```yaml
# Retrieve secrets from Vault
config_sources:
  vault:
    # The address of your Vault server (or use env var VAULT_ADDR)
    endpoint: "http://localhost:8200"

    # The path to your secrets engine
    path: "secret/data/kv"

    # How often to check for updates (for KV v2 polling)
    poll_interval: 90s

    # Authentication configuration
    auth:
      token: "${env:VAULT_TOKEN}"

......
exporters:
  otlp:
    # Use 'otlp.eu01.nr-data.net:4317' for New Relic EU accounts
    endpoint: "otlp.nr-data.net:4317"
    headers:
      "api-key": "${vault:data.nr-api-key}"
......
```

Best Practices

Detached Deployment in Nomad

```
nomad job run -detach otel-demo-app/jobspec/otel-collector.nomad
```

This prevents the deployment pipeline from hanging if one non-critical service (like a load generator) takes longer to start. It ensures the observability "substrate" comes up independently of the application logic

Best Practices

Using memory_limiter

```yaml
processors:
  batch:
  memory_limiter:
    check_interval: 1s          # Check memory every second [2]
    limit_mib: 1024             # Hard limit for the process [2]
    spike_limit_percentage: 20  # 20% buffer for sudden bursts [2]
```

Collector accumulates data faster than it can export it, its memory usage will skyrocket in milliseconds. If it hits the container's memory limit (OOM), the orchestrator (Kubernetes/Nomad) will kill the process, resulting in lost observability data

ROI of Unified Observability

Faster RCA

Eliminate the blame game and correlate a “Load Balancer 502” error to a specific Terraform update

Collaborations

Dev and Ops can share the unified observability and eliminate “it works on my machine”

Continuous Drift Detection

Alerts on unauthorized changes where state diverages from code

THANK YOU