Log Management & Analytics

Powered by Dynatrace Grail™

Observability is growing fast

The importance of observability is increasing.
Use cases are more diverse.
Observe everything is the mindset.

Infrastructure monitoring

Hybrid cloud observability

Fault domain isolation

Serverless, container, pod & network

Applications & microservices

Hybrid cloud distributed tracing

Automatic code-level root-cause and profiling

Front & back-end availability and performance

Digital experience

Mobile, web browser and API

Feature adoption analysis

Visual session replay

Application Security

Run-time vulnerability detection

Impact analysis

DevSecOps Automation

Cloud automation

Closed-loop remediation

Quality gate, service level and delivery

DevOps, SRE lifecycle

Digital business

Real-time business insights

Impact and conversion

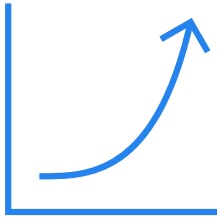
BizDevOps integration and automation

There’s a problem

The volume, variety and velocity of data will exceed the scalability of the observability solutions.

Finding answers and root-cause will be impossible with human-powered correlation. Automated answers and causation is required.

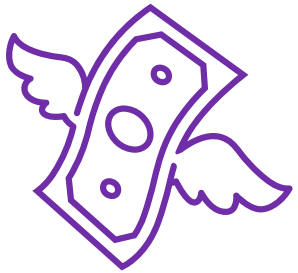
Problems with state-of-the-art technology



Data explosion

Multi-cloud environment, serverless architecture, … results in vast increase of data.

How can I address the data ingest barrier?



Increasing costs

More data means higher costs! ***Can I afford data retention?***

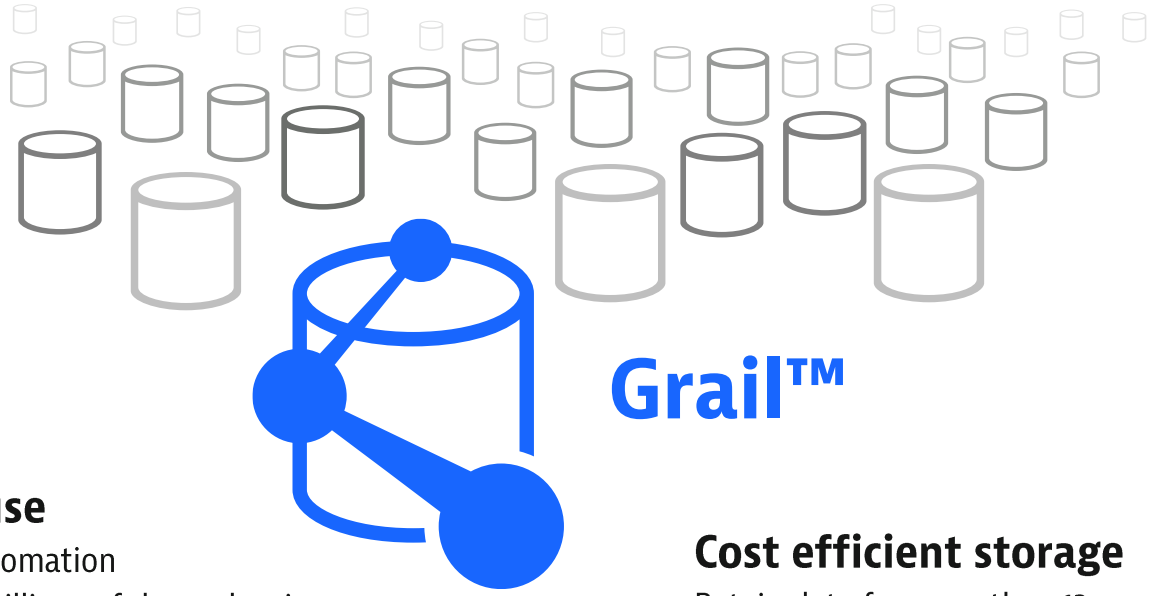
Why do I have to decide which data will be relevant for my business?

Cost / value pain

I need better analytics! *Which insights do I get for (paid) data?*

"Rehydration" is inefficient and slow! *How to manage data?*

Next Gen Analytics Engine



Purpose-built data lakehouse

Focus on observability, security and automation
Optimized for Dynatrace AI to process billions of dependencies

Unified storage for observability data

Single data store for metrics, traces, logs and more
Data stored in context within real-time dependency model

>100x more scalable

100 TB/day data ingest per tenant (Q1 CY23)
Aiming for 1000 PB/day depending on future market needs

Cost efficient storage

Retain data for more than 12 months
No need to manage cold, warm or hot storage tiers

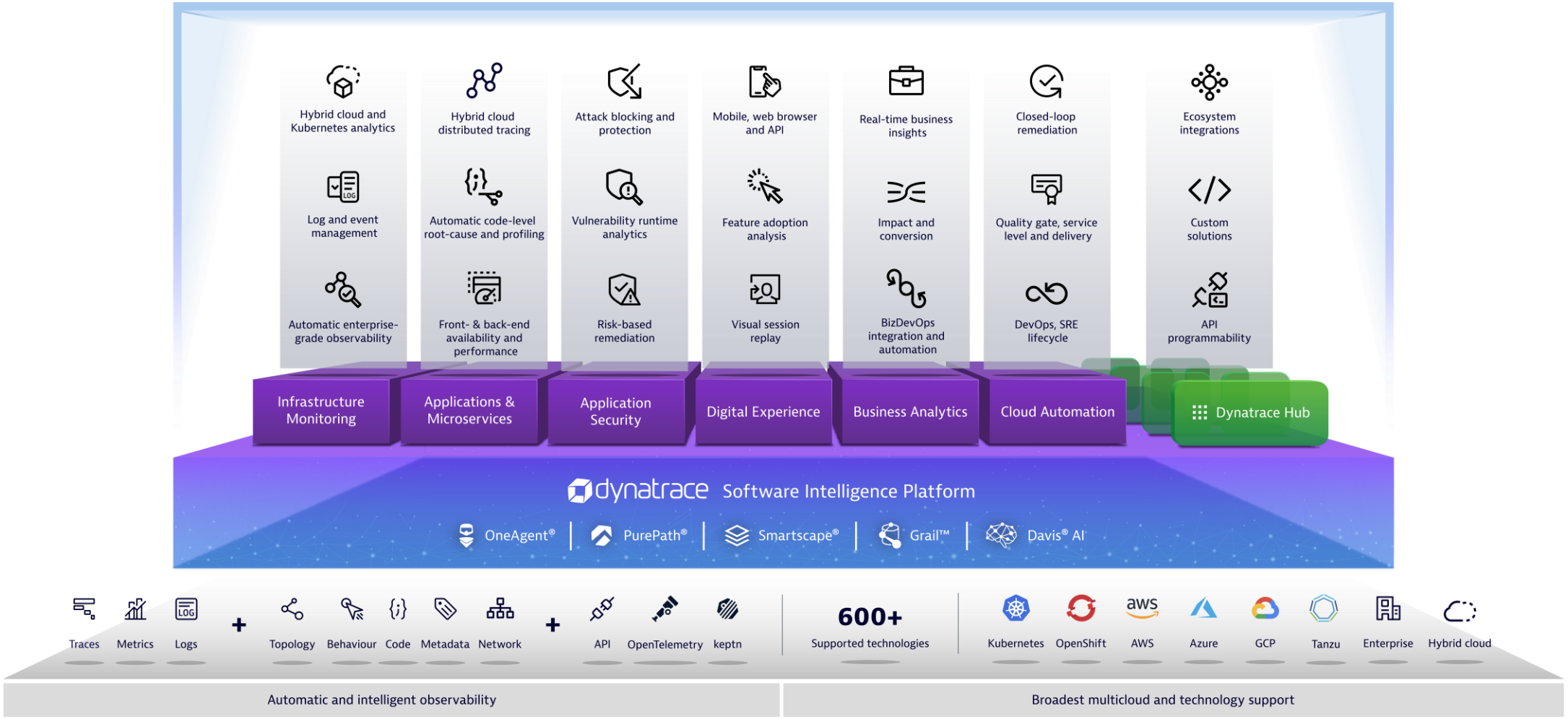
Instant answers

No index, no schema definition, no rehydration
Data analytics at any time with Dynatrace Query Language

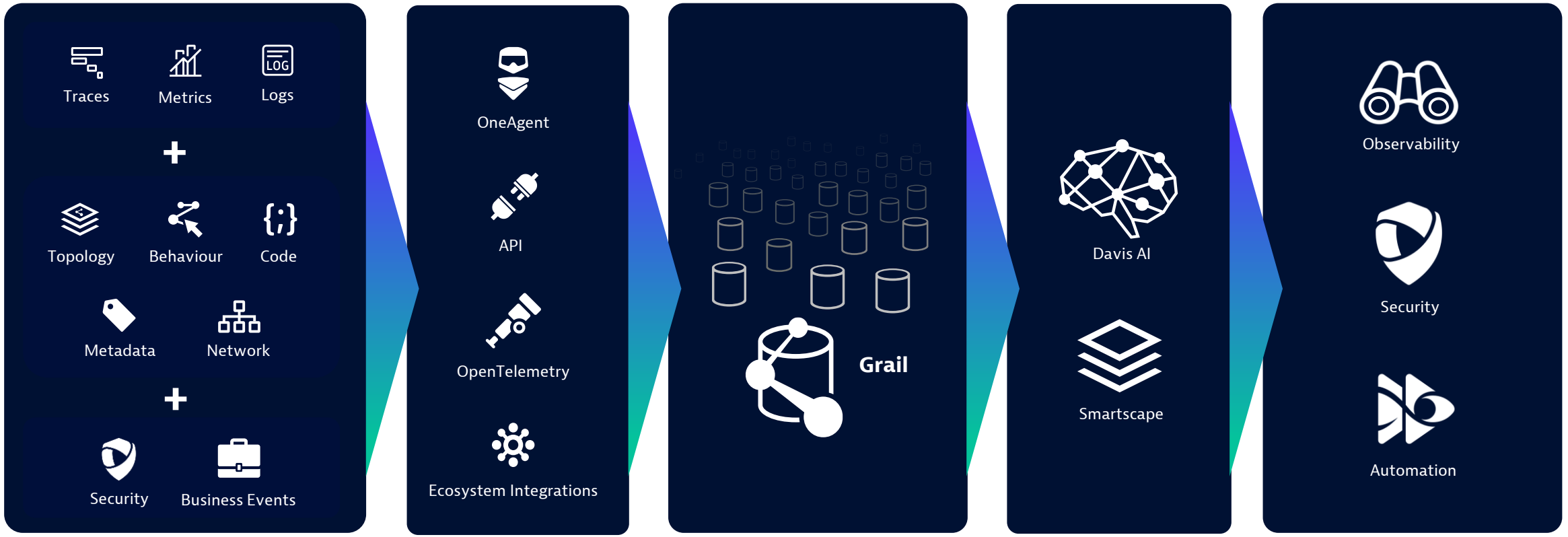
Query performance up to 100x faster

Massive parallel processing on 1000s of nodes
~1TB in ~1s @ 1000 cores

Grail core technology powers all platform modules



Deliver answers and intelligent automation from data



Deep, context rich, full stack data sources beyond observability

Automatically captured with topology context & pre-processing

Data storage with massive processing & retrieval capabilities

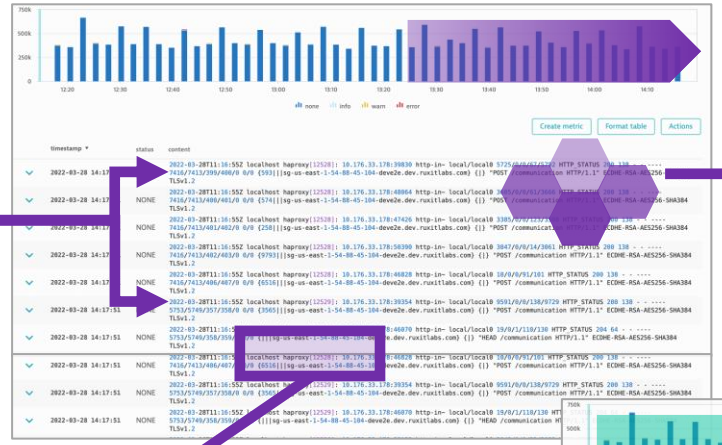
Causal AI analysis & answers

Powering observability, security & automation use cases

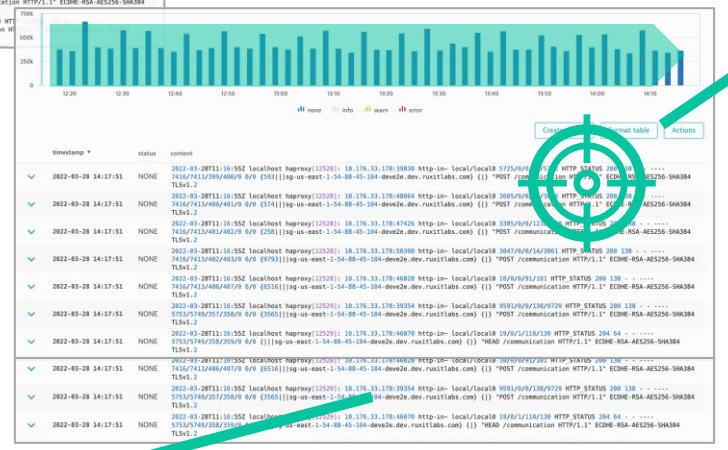
Logs powered by Grail – Advantages over industry standard

Indexed Database

1. Decide relevant indexes before ingest
2. Limited query operators
3. Simple keyword search
4. Wait for new logs to arrive



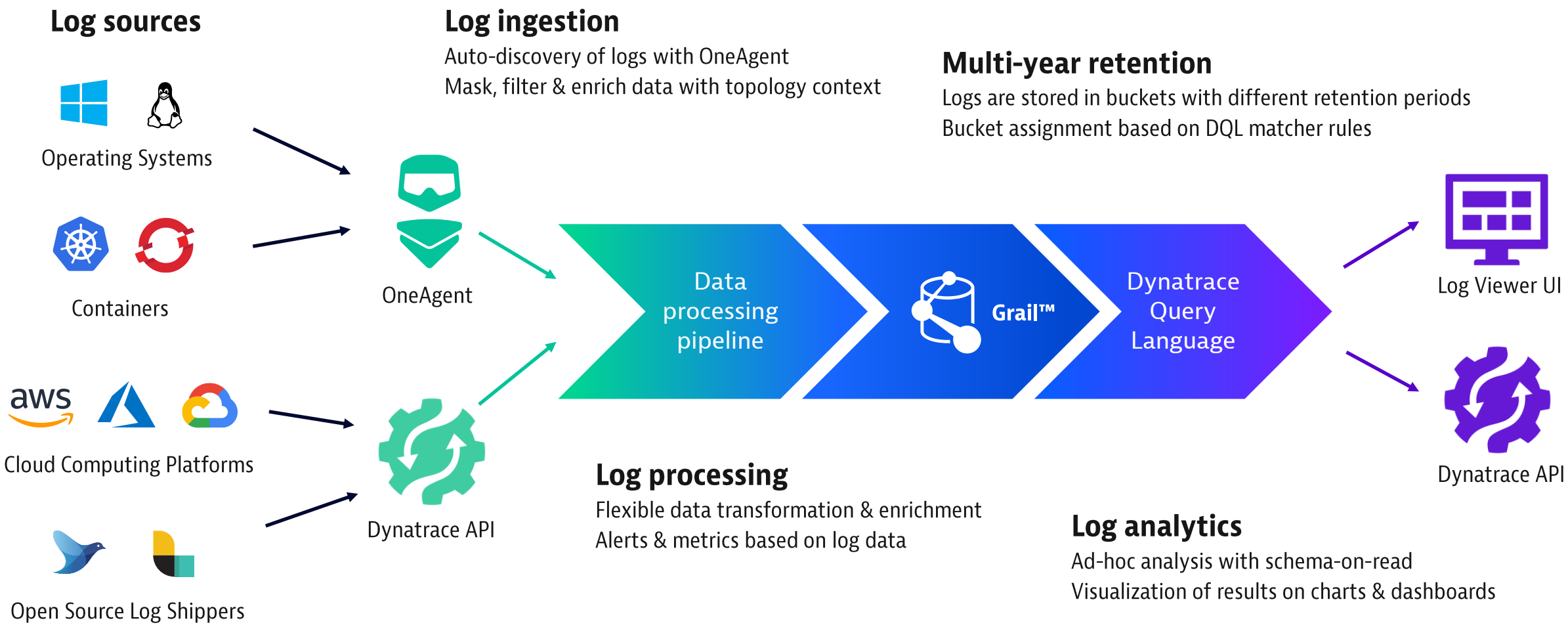
1. Decide what is relevant in the logs at any time
2. Powerful data exploration & transformation with DQL
3. Unlimited transforming, restructuring of all raw logs



...vs Grail™

4. Full historical analysis
5. Statistical approximate (sampled) summaries
5. 100% precision insight into all data stored

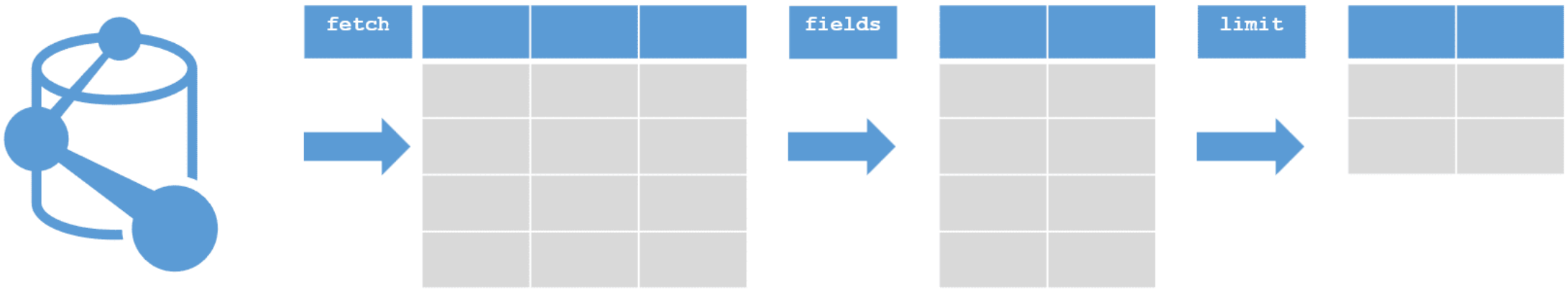
Log Management & Analytics – Architecture



Dynatrace Query Language

```
fetch logs
| filter loglevel == "SEVERE" or loglevel == "ERROR"
| summarize count = count(), by:{bin(timestamp, 5m), loglevel}
| sort count desc
```

Chaining commands with the pipe operator

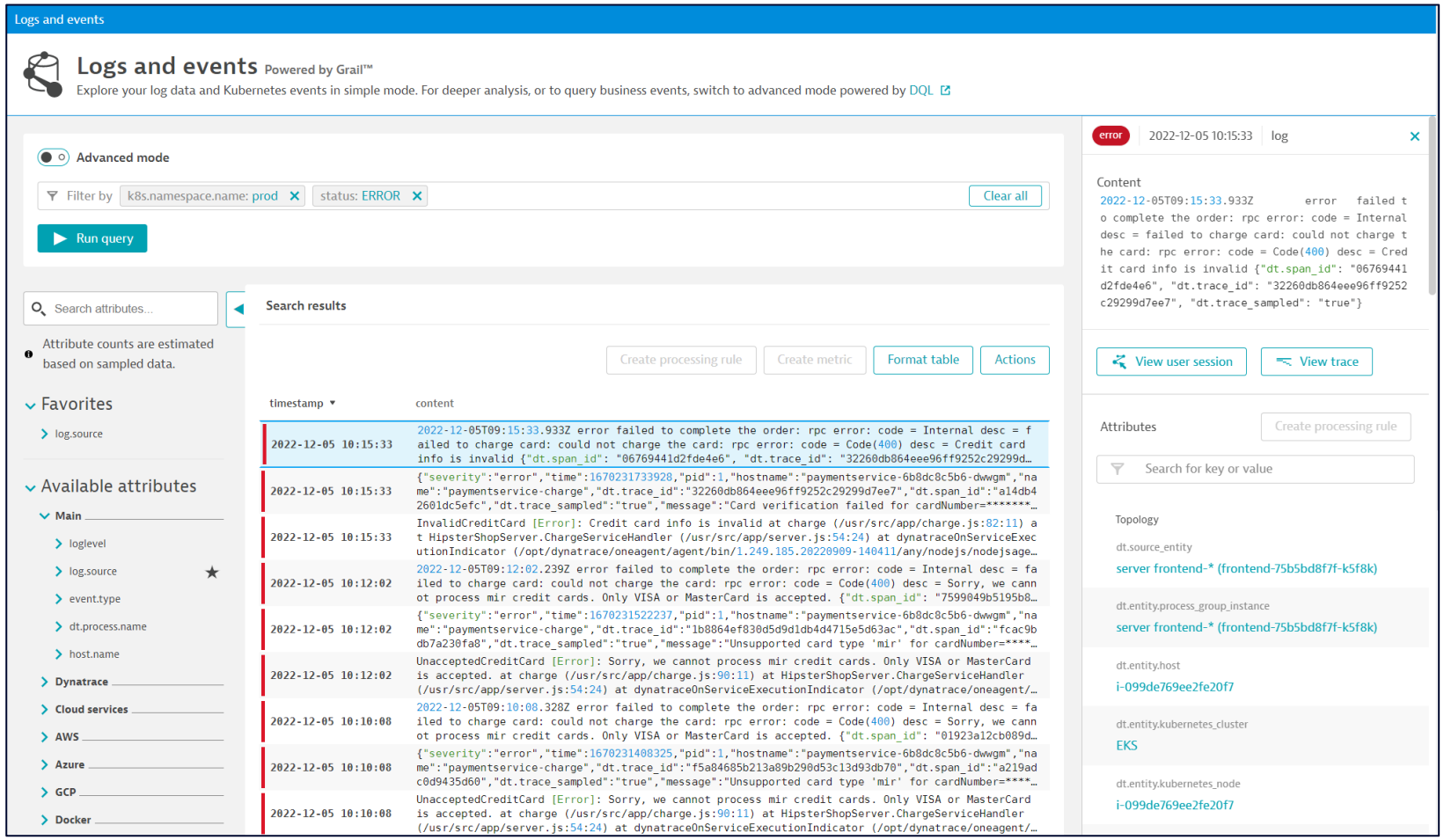


DQL Example

```
fetch       logs, from:now()-20m
| filter    endsWith(log.source,"/media/datastore/server-data/log/audit.config.change.#.#.log")
            and dt.host_group.id == "cluster_deve2e"
| parse     content, "timestamp('yyyy-MM-dd HH:mm:ss'):ts
                      ld json:settings
                      ipaddr:client_ip //IPv4/6"
| fields    ts,
            type = settings[eventType],
            tenant = settings[tenantId],
            user = settings[userId],
            change = settings[jsonPatch]
| filter    in(type,array("UPDATE","DELETE")) and user != "unknown"
| summarize creates = countIf(type=="CREATE"), upd = countIf(type=="UPDATE"), del = countIf(type=="DELETE"),
            by:{tenant, user}
| fieldsAdd changes_per_min = (upd + del)/20
| sort      changes_per_min desc
```

Demo

Log Viewer



Simple mode

For troubleshooting use cases

Quick search

Filter by context attributes

Context-aware

Log records with topology context
e.g. host, application, K8s namespace

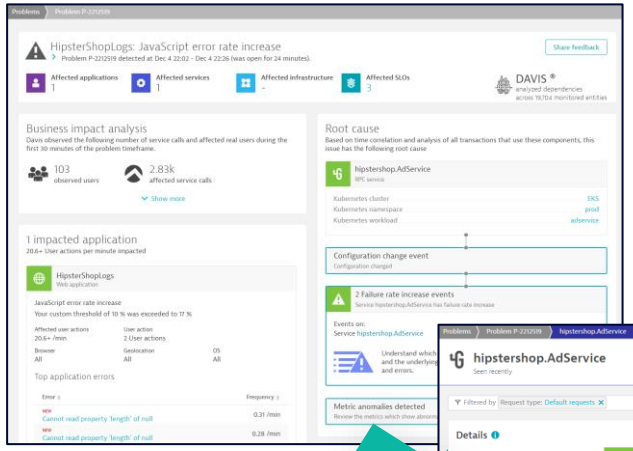
Connected details

Drill down to linked entities, distributed traces and user sessions

Troubleshooting with logs in context

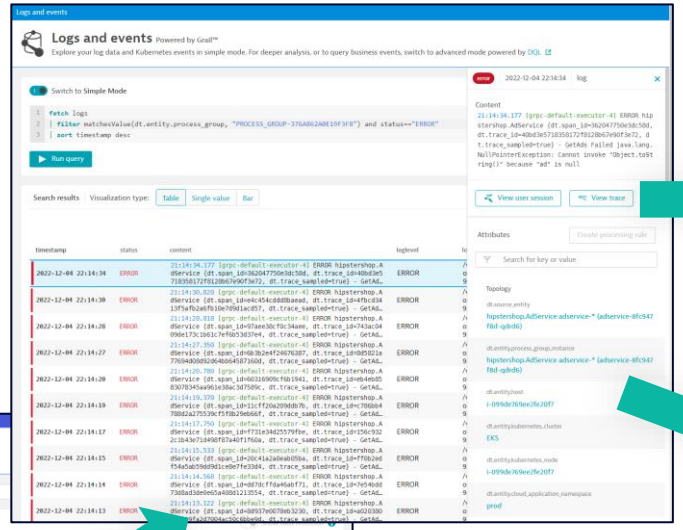
Anomaly detection

Dynatrace auto-detects application problem with increased error rate



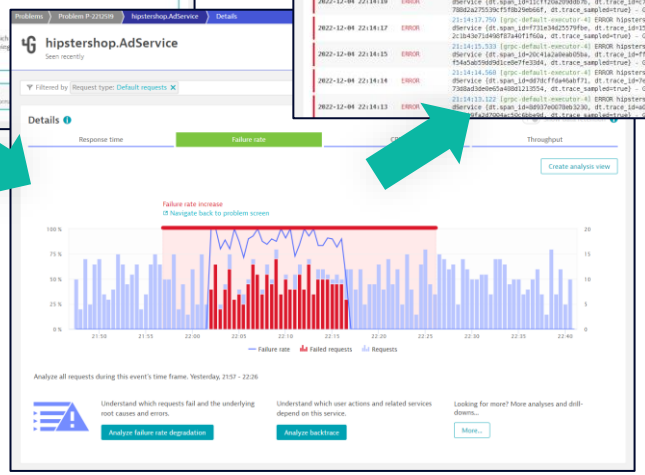
Drill down to logs

Analyze related error logs to identify issue



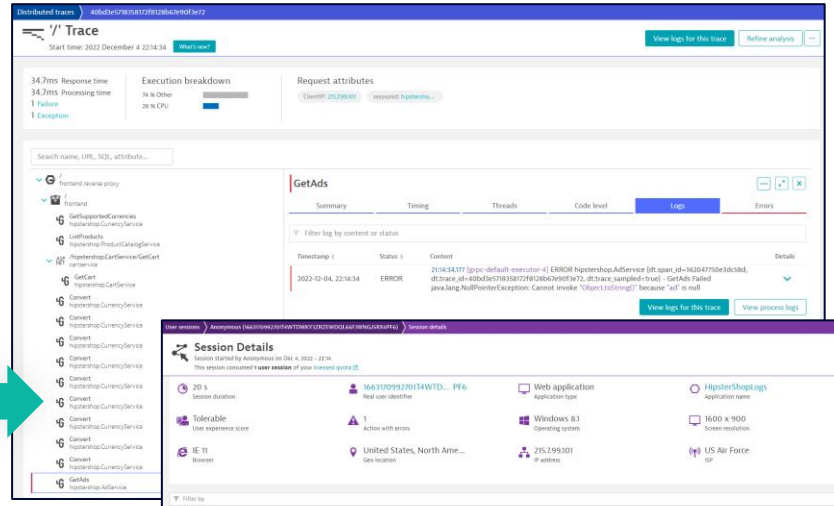
Root cause analysis

Problem caused by backend service with failing requests



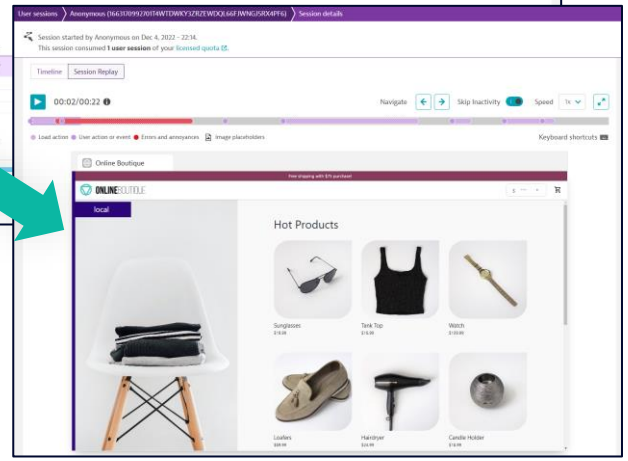
Logs linked to distributed traces

Check logs in context of the end-to-end transaction flow



Session replay

Understand which logs really affect the user experience of connected session and how logs have been created



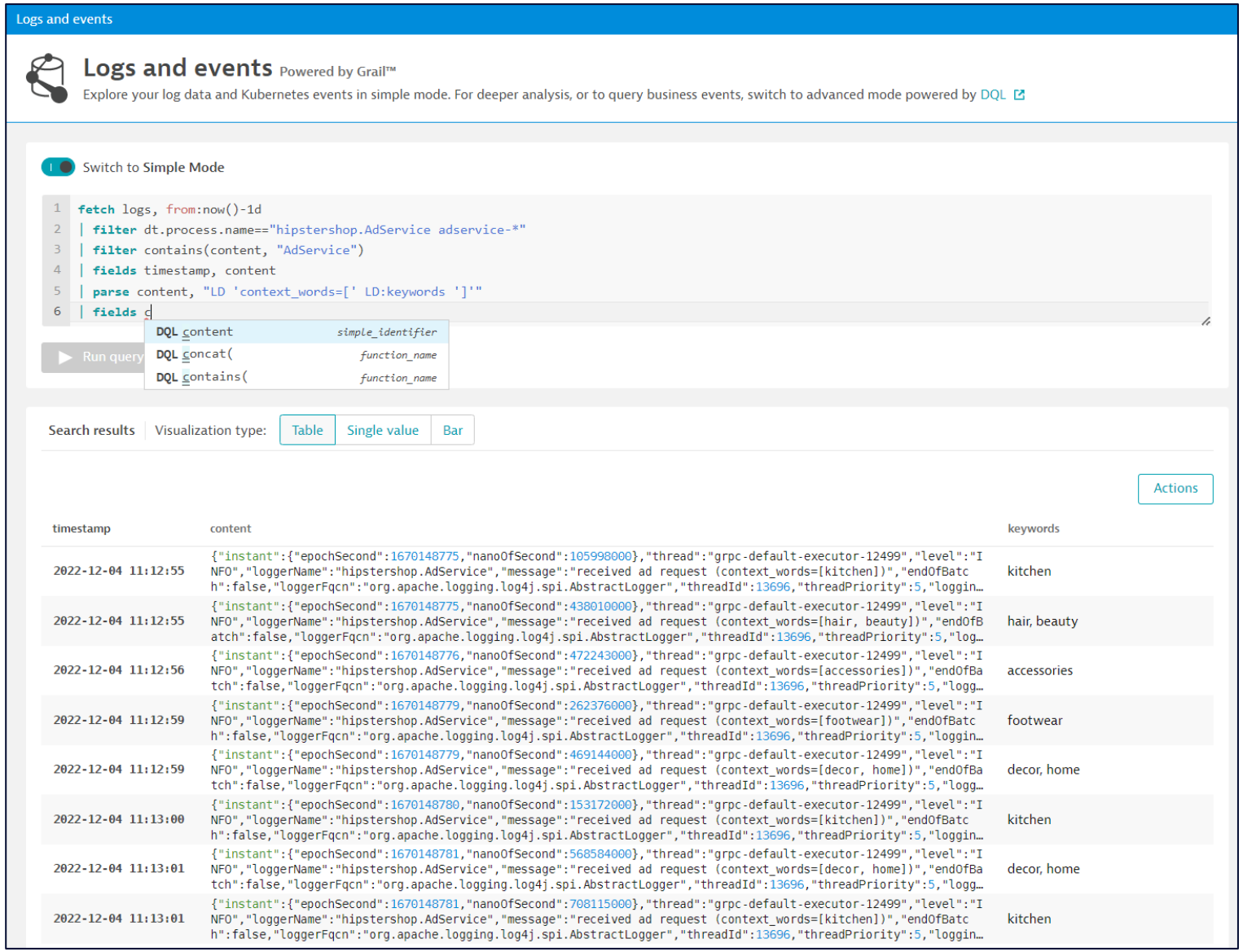
Log Viewer

Advanced mode
For log analytics use cases

Query editor
Full control over result with DQL
Autocomplete support

Instant answers
Filter, parse and aggregate log data to create reports at any time

Visualization
Turn results into charts
Pin charts to dashboards



Advanced log use cases with DQL

Deployment verification – Find failure traces logged more than 10 times within a 5-minute interval.

```
fetch logs
| filter loglevel=="ERROR" and k8s.namespace.name=="prod“
| summarize count = count(), by:{bin(timestamp, 5m), dt.process.name}
| filter count > 10
| sort count desc
```

Business analytics – What are the most popular products in our online store?

```
fetch logs, from:now()-3h
| filter dt.process.name=="cartservice cartservice-*"
| filter contains(content, "AddItemAsync")
| parse content, "LD 'userId=' LD:userId ', productId=' LD:productId ', quantity=' INT:productQuantity"
| fields productId , productQuantity
| summarize averageProductQuantity = avg(productQuantity), by:{productId}
| sort averageProductQuantity desc
| limit 5
```

Fault isolation – Are the error logs correlated with too many requests from the same IP?

```
fetch logs
| filter status=="ERROR“
| parse content, "LD, IPADDR:ip“
| summarize errorcount = count(), by:{ip}
| sort errorcount desc
```

Audit & forensics – Is that a malicious user? Check the audit logs for user input patterns for last 12 months.

```
fetch logs, from:now()-1y
| filter endsWith(log.source, "audit.log") and contains(content, "Unknown user name or bad password")
| parse content, "DATA 'Failed:' DATA 'Account Name:' BLANK LD?:’userId' EOLWIN DATA 'Account Domain:' BLANK LD?:’accountDomain' EOLWIN"
| summarize loginAttempts=count(), by:{bin(timestamp,1d), userId}
| sort attempts desc
```