

Log Management & Analytics

Powered by Dynatrace Grail™

The background features a blue-to-teal gradient. In the top-left and bottom-right corners, there are white geometric patterns of interconnected lines forming triangles and diamonds.

Observability is growing fast

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



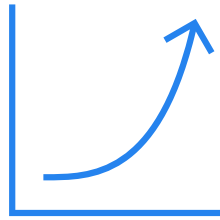
There's a problem

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.



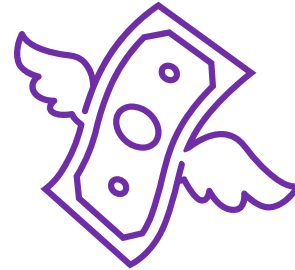
Problems with state-of-the-art technology



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?

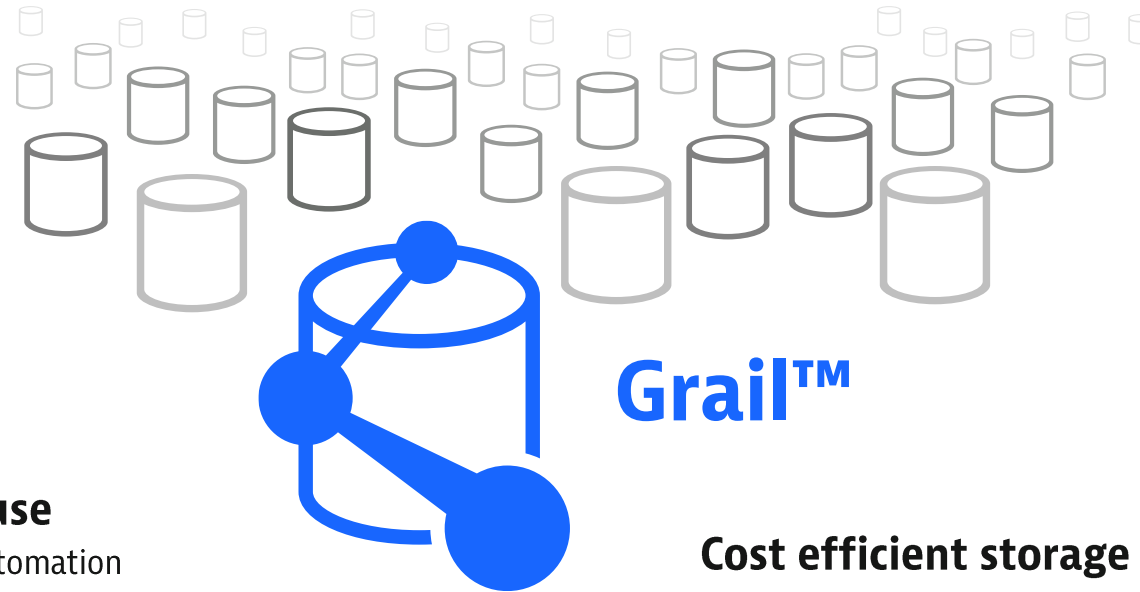


Cost / value pain

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

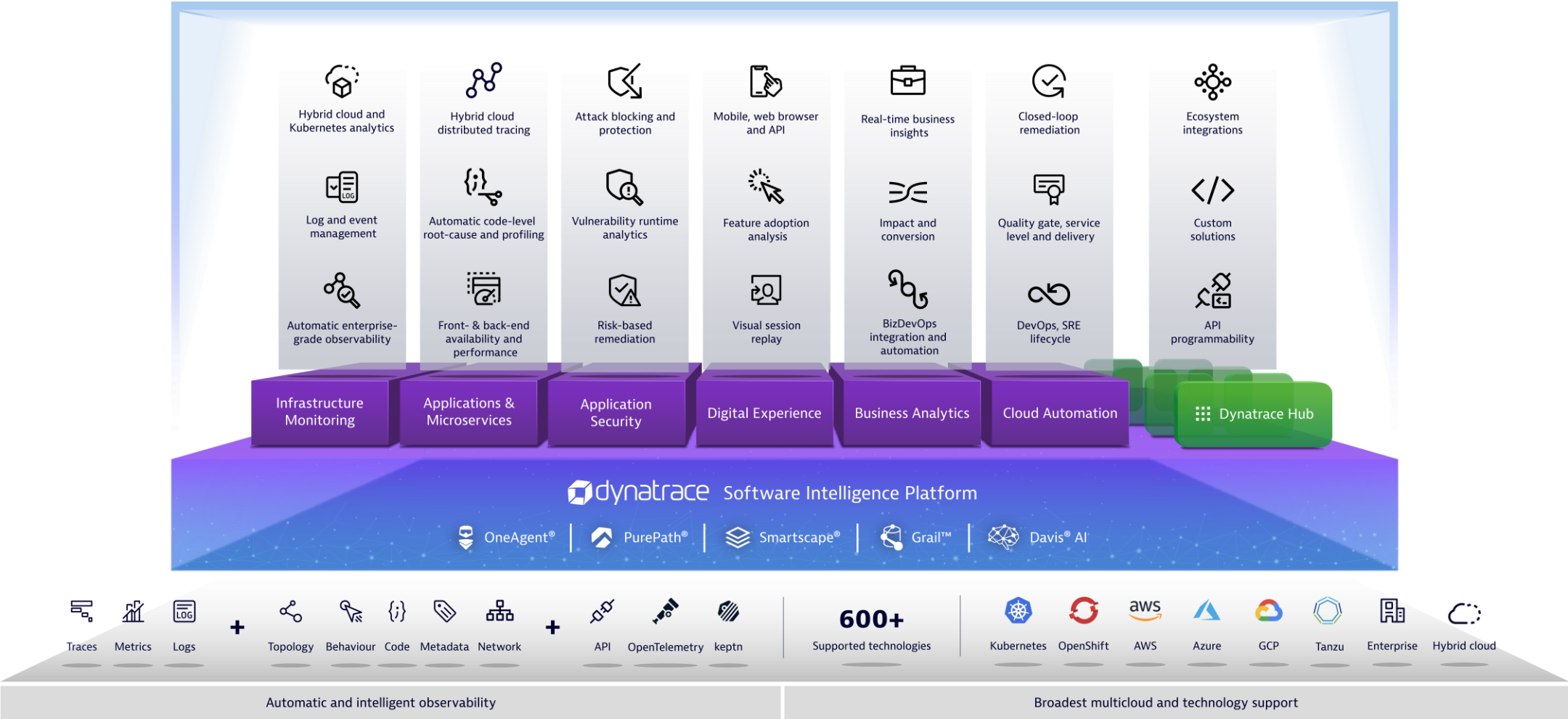
Instant answers

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

Traces
Metrics
Logs

+

Topology
Behaviour
Code

Metadata
Network

+

Security
Business Events

Deep, context rich, full stack data sources beyond observability

OneAgent

API

OpenTelemetry

Ecosystem Integrations

Automatically captured with topology context & pre-processing

Grail

Data storage with massive processing & retrieval capabilities

Davis AI

Smartscape

Causal AI analysis & answers

Observability

Security

Automation

Powering observability, security & automation use cases



Logs powered by Grail – Advantages over industry standard

...vs Grail™

1 Decide relevant indexes before ingest

2 Limited query operators

3 Simple keyword search

Indexed Database

1 Decide what is relevant in the logs at any time

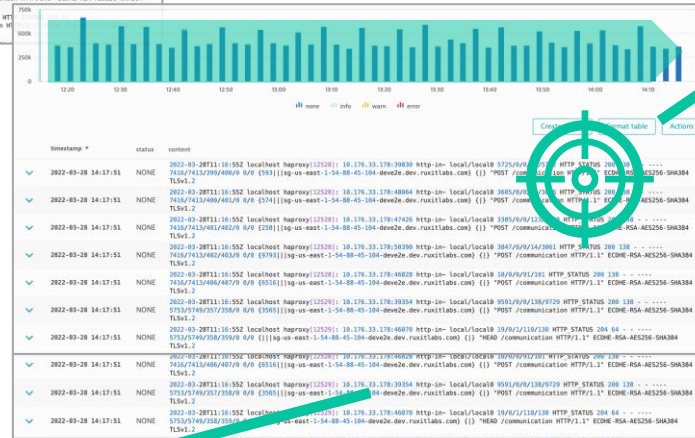
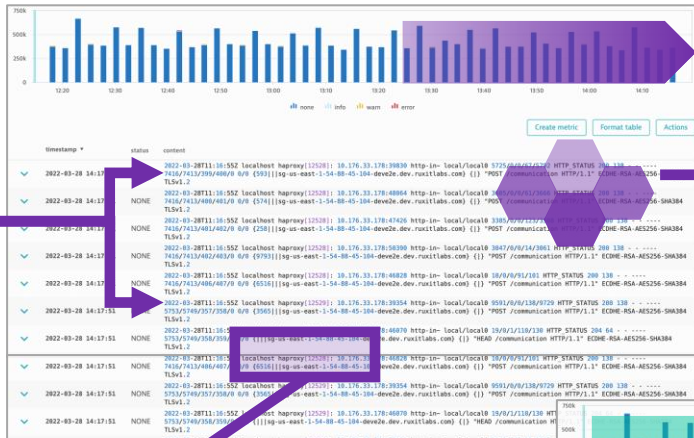
3 Unlimited transforming, restructuring of all raw logs

4 Wait for new logs to arrive

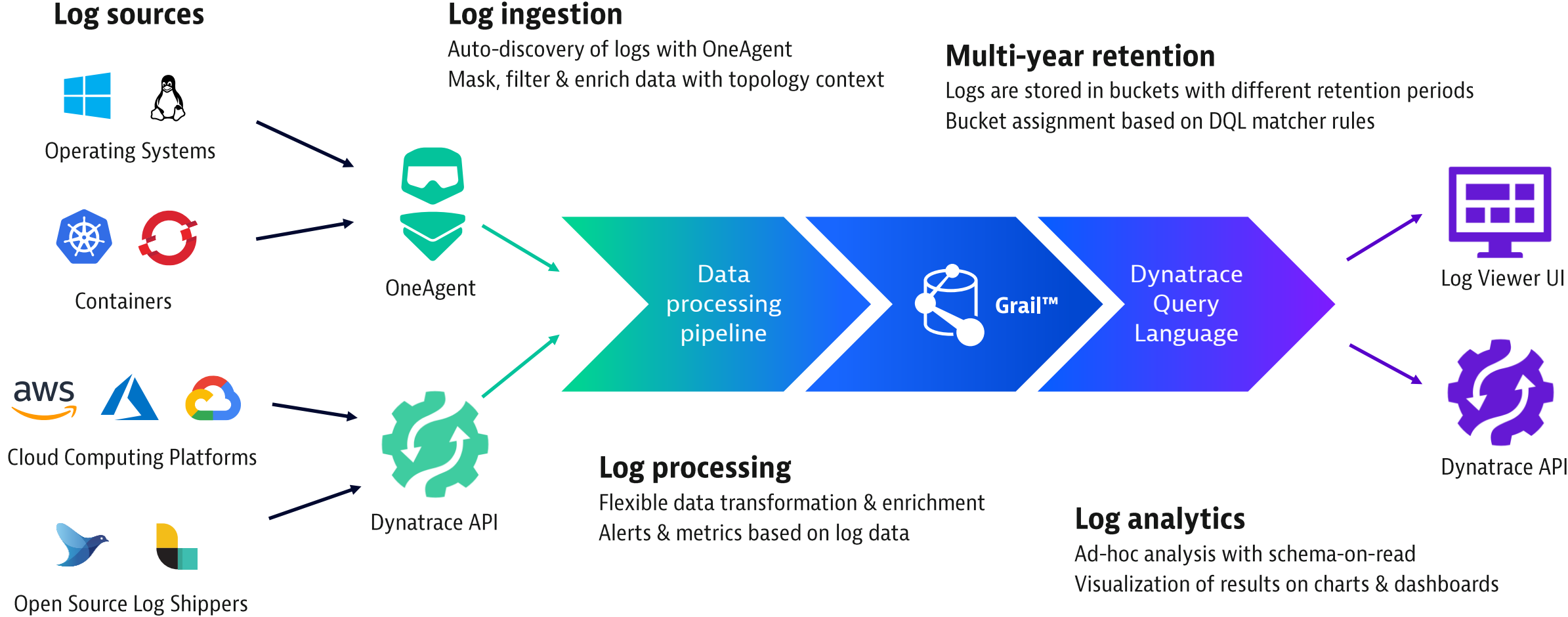
5 Statistical approximate (sampled) summaries

4 Full historical analysis

5 100% precision insight into all data stored



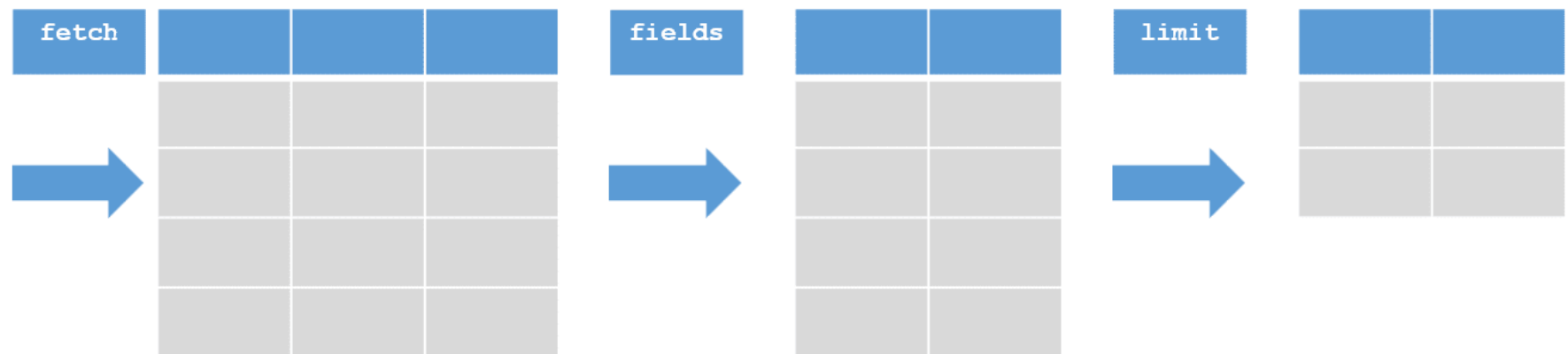
Log Management & Analytics – Architecture



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

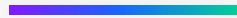


DQL Example

```
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

Logs and events Powered by Grail™
Explore your log data and Kubernetes events in simple mode. For deeper analysis, or to query business events, switch to advanced mode powered by DQL

Advanced mode

Filter by `k8s.namespace.name: prod` `status: ERROR` [Clear all](#)

[Run query](#)

Search attributes... Search results

Attribute counts are estimated based on sampled data.

Favorites

- log.source

Available attributes

- Main
 - log.level
 - log.source
 - event.type
 - dt.process.name
 - host.name
- Dynatrace
- Cloud services
 - AWS
 - Azure
 - GCP
 - Docker

| timestamp | content |
|---------------------|---|
| 2022-12-05 10:15:33 | 2022-12-05T09:15:33.933Z error failed to complete the order: rpc error: code = Internal desc = failed to charge card: could not charge the card: rpc error: code = Code(400) desc = Credit card info is invalid {"dt.span_id": "06769441d2fde4e6", "dt.trace_id": "32260db864eee96ff9252c29299d..."} |
| 2022-12-05 10:15:33 | {"severity": "error", "time": "1670231733928", "pid": "1", "hostname": "paymentservice-6b8dc8c5b6-dwgm", "name": "paymentservice-charge", "dt.trace_id": "32260db864eee96ff9252c29299d7ee7", "dt.span_id": "a14db42601dc5efc", "dt.trace_sampled": "true", "message": "Card verification failed for cardNumber=*****"} |
| 2022-12-05 10:15:33 | InvalidCreditCard [Error]: Credit card info is invalid at charge (/usr/src/app/charge.js:92:11) at HipsterShopServer.ChargeServiceHandler (/usr/src/app/server.js:54:24) at dynatraceOnServiceExecutionIndicator (/opt/dynatrace/oneagent/agent/bin/1.249.185.20220909-140411/any/nodejs/nodejsage... |
| 2022-12-05 10:12:02 | 2022-12-05T09:12:02.239Z error failed to complete the order: rpc error: code = Internal desc = failed to charge card: could not charge the card: rpc error: code = Code(400) desc = Sorry, we cannot process mir credit cards. Only VISA or MasterCard is accepted. {"dt.span_id": "7599049b5195b8..."} |
| 2022-12-05 10:12:02 | {"severity": "error", "time": "1670231522237", "pid": "1", "hostname": "paymentservice-6b8dc8c5b6-dwgm", "name": "paymentservice-charge", "dt.trace_id": "1b8864ef830d5d9d1db4d4715e5d63ac", "dt.span_id": "fcac9bcb7a230fa8", "dt.trace_sampled": "true", "message": "Unsupported card type 'mir' for cardNumber=*****"} |
| 2022-12-05 10:12:02 | UnacceptedCreditCard [Error]: Sorry, we cannot process mir credit cards. Only VISA or MasterCard is accepted. at charge (/usr/src/app/charge.js:90:11) at HipsterShopServer.ChargeServiceHandler (/usr/src/app/server.js:54:24) at dynatraceOnServiceExecutionIndicator (/opt/dynatrace/oneagent/... |
| 2022-12-05 10:10:08 | 2022-12-05T09:10:08.328Z error failed to complete the order: rpc error: code = Internal desc = failed to charge card: could not charge the card: rpc error: code = Code(400) desc = Sorry, we cannot process mir credit cards. Only VISA or MasterCard is accepted. {"dt.span_id": "01923a12cb089d..."} |
| 2022-12-05 10:10:08 | {"severity": "error", "time": "1670231488325", "pid": "1", "hostname": "paymentservice-6b8dc8c5b6-dwgm", "name": "paymentservice-charge", "dt.trace_id": "f5a84685b213a89b290d53c13d93db70", "dt.span_id": "a219adc0d9435d60", "dt.trace_sampled": "true", "message": "Unsupported card type 'mir' for cardNumber=*****"} |
| 2022-12-05 10:10:08 | UnacceptedCreditCard [Error]: Sorry, we cannot process mir credit cards. Only VISA or MasterCard is accepted. at charge (/usr/src/app/charge.js:90:11) at HipsterShopServer.ChargeServiceHandler (/usr/src/app/server.js:54:24) at dynatraceOnServiceExecutionIndicator (/opt/dynatrace/oneagent/... |

[View user session](#) [View trace](#)

Attributes [Create processing rule](#)

Search for key or value

Topology

- dt.source_entity
 - server frontend-* (frontend-75b5bd8f7f-k5f8k)
- dt.entity.process_group.instance
 - server frontend-* (frontend-75b5bd8f7f-k5f8k)
- dt.entity.host
 - i-099de769ee2fe20f7
- dt.entity.kubernetes_cluster
 - EKS
- dt.entity.kubernetes_node
 - i-099de769ee2fe20f7

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

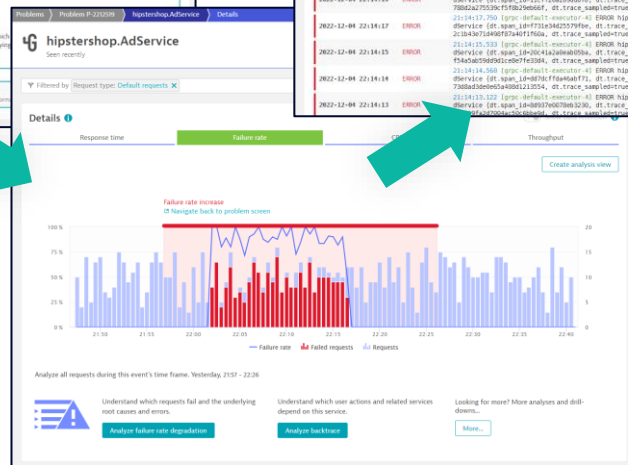
Anomaly detection

Dynatrace auto-detects application problem with increased error rate

The screenshot shows a problem titled "HipstersShop: JavaScript error rate increase" detected on Dec 4, 2022. It lists affected applications, services, and infrastructure. A "Business impact analysis" section shows 103 observed users and 2,836 affected service calls. A "Root cause" section identifies "hipstershop.AdService" as the source, with a "Configuration change event" noted. A "1 impacted application" section shows the JavaScript error rate increase. A "Top application errors" table lists errors like "Cannot read property 'length' of null".

Root cause analysis

Problem caused by backend service with failing requests



Drill down to logs

Analyze related error logs to identify issue

The screenshot shows the "Logs and events" interface. It includes a search bar, a "Run query" button, and a table of search results. The table columns are "timestamp", "status", "content", and "loglevel". The content column shows error messages from the "hipstershop.AdService" component, such as "ERROR hipstershop.AdService [et:span_id=904779693d368, dt:trace_id=4043637838537782326c9c6f6d7a72, d:1, trace_sampled=true] - GetAds Failed: java.lang.NullPointerException: Cannot invoke 'Object.toString()' because 'sp' is null".

Logs linked to distributed traces

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

The screenshot shows the "Distributed traces" interface for a "Trace" of type "GetAds". It displays a "Request attributes" table, an "Execution breakdown" bar chart, and a "Log" section. The log section shows a detailed view of the error log from the previous screenshot, including the full stack trace and the associated trace ID.

The screenshot shows the "Session Details" interface for a session identified by "186181999270114WTD...". It provides information about the session duration (20 s), user experience score (Tolerable), and user location (United States, North America). It also lists the application (HipstersShop), operating system (Windows R1), and IP address (275.259.101). A "Timeline" section shows the sequence of events during the session.

Session replay

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

The screenshot shows the "Session replay" interface, displaying a user session from an anonymous user. It includes a "Timeline" section with a play button and a "Session replay" section showing a sequence of user actions. Below this, there is a "User session" section showing a screenshot of the "Online Boutique" website, with "Hot Products" like sunglasses, tank tops, and shoes.

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

Logs and events

Powered by Grail™
Explore your log data and Kubernetes events in simple mode. For deeper analysis, or to query business events, switch to advanced mode powered by DQL

Switch to Simple Mode

```
1 fetch logs, from:now()-1d
2 | filter dt.process.name=="hipstershop.AdService adservice-*"
3 | filter contains(content, "AdService")
4 | fields timestamp, content
5 | parse content, "LD 'context_words=[ LD:keywords ]'"
6 | fields d
```

Run query

| | |
|-----------------------|-------------------|
| DQL <u>concat</u> | simple_identifier |
| DQL <u>concat</u> (| function_name |
| DQL <u>contains</u> (| function_name |

Search results Visualization type: Actions

| timestamp | content | keywords |
|---------------------|--|--------------|
| 2022-12-04 11:12:55 | {"instant":{"epochSecond":1670148775,"nanoOfSecond":105998000},"thread":"grpc-default-executor-12499","level":"INFO","loggerName":"hipstershop.AdService","message":"received ad request (context_words=[kitchen])","endOfBatch":false,"loggerFqcn":"org.apache.logging.log4j.spi.AbstractLogger","threadId":13696,"threadPriority":5,"log... | kitchen |
| 2022-12-04 11:12:55 | {"instant":{"epochSecond":1670148775,"nanoOfSecond":438010000},"thread":"grpc-default-executor-12499","level":"INFO","loggerName":"hipstershop.AdService","message":"received ad request (context_words=[hair, beauty])","endOfBatch":false,"loggerFqcn":"org.apache.logging.log4j.spi.AbstractLogger","threadId":13696,"threadPriority":5,"log... | hair, beauty |
| 2022-12-04 11:12:56 | {"instant":{"epochSecond":1670148776,"nanoOfSecond":472243000},"thread":"grpc-default-executor-12499","level":"INFO","loggerName":"hipstershop.AdService","message":"received ad request (context_words=[accessories])","endOfBatch":false,"loggerFqcn":"org.apache.logging.log4j.spi.AbstractLogger","threadId":13696,"threadPriority":5,"log... | accessories |
| 2022-12-04 11:12:59 | {"instant":{"epochSecond":1670148779,"nanoOfSecond":262376000},"thread":"grpc-default-executor-12499","level":"INFO","loggerName":"hipstershop.AdService","message":"received ad request (context_words=[footwear])","endOfBatch":false,"loggerFqcn":"org.apache.logging.log4j.spi.AbstractLogger","threadId":13696,"threadPriority":5,"loggin... | footwear |
| 2022-12-04 11:12:59 | {"instant":{"epochSecond":1670148779,"nanoOfSecond":469144000},"thread":"grpc-default-executor-12499","level":"INFO","loggerName":"hipstershop.AdService","message":"received ad request (context_words=[decor, home])","endOfBatch":false,"loggerFqcn":"org.apache.logging.log4j.spi.AbstractLogger","threadId":13696,"threadPriority":5,"log... | decor, home |
| 2022-12-04 11:13:00 | {"instant":{"epochSecond":1670148780,"nanoOfSecond":153172000},"thread":"grpc-default-executor-12499","level":"INFO","loggerName":"hipstershop.AdService","message":"received ad request (context_words=[kitchen])","endOfBatch":false,"loggerFqcn":"org.apache.logging.log4j.spi.AbstractLogger","threadId":13696,"threadPriority":5,"loggin... | kitchen |
| 2022-12-04 11:13:01 | {"instant":{"epochSecond":1670148781,"nanoOfSecond":568584000},"thread":"grpc-default-executor-12499","level":"INFO","loggerName":"hipstershop.AdService","message":"received ad request (context_words=[decor, home])","endOfBatch":false,"loggerFqcn":"org.apache.logging.log4j.spi.AbstractLogger","threadId":13696,"threadPriority":5,"log... | decor, home |
| 2022-12-04 11:13:01 | {"instant":{"epochSecond":1670148781,"nanoOfSecond":708115000},"thread":"grpc-default-executor-12499","level":"INFO","loggerName":"hipstershop.AdService","message":"received ad request (context_words=[kitchen])","endOfBatch":false,"loggerFqcn":"org.apache.logging.log4j.spi.AbstractLogger","threadId":13696,"threadPriority":5,"loggin... | kitchen |



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
```

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
```

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
```

