



PERFORM

INTRO TO DQL



Perform 2024

HANDS-ON TRAINING // powered by 



**SANTIAGO PALACIOS
GARCIA**

Director Solutions
Engineering



JAKE NORKUS

Solutions Engineer



ALEX BREWERS

Solutions Engineer



JASON OSTROSKI

Principal Solutions
Engineer

AGENDA



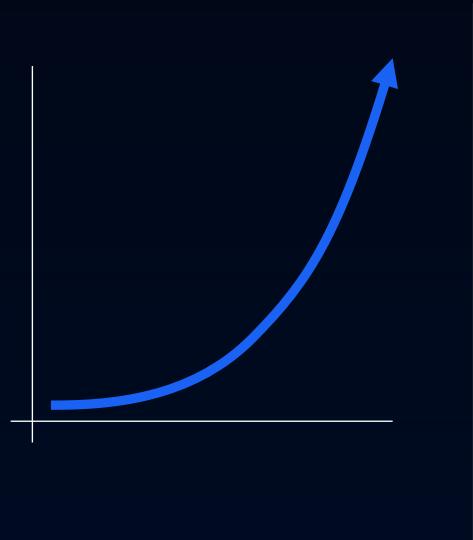
- Introduction to Grail – Dynatrace's Data Lakehouse for unified observability and security
- What is the Dynatrace Query Language (DQL) and where can it be used in Dynatrace?
- DQL Exercises with Logs
- DQL Exercises with BizEvents
- Break
- DQL Exercises with Metrics
- DQL Exercises with Events and Topology
- Davis Copilot Exercises
- Additional learning resources

WHY DID WE BUILD GRAIL?



Macro trends show that current strategies can't keep up

Data collection
explosion



Data ingest grows
significantly



Observability + security,
automation convergence



Siloed tooling without
context



WE DELIVER **ANSWERS** AND
INTELLIGENT AUTOMATION FROM **DATA**



PERFORM



GRAIL™

LAUNCHED in OCT 2022

Causalational data lakehouse,
massively parallel processing,
context-rich analytics



PERFORM

DELIVERING ANSWERS AND INTELLIGENT AUTOMATION FROM DATA



Grail™

Purpose-built data lakehouse

Focus on observability, security and automation

Optimized for Dynatrace AI to process billions of dependencies

Observability data in full context

Single datastore for all observability signals, with real-time topology mapping

Logs and traces, fully connected to get answers fast

100x more scalable

250 TB/day data ingest per tenant and growing!

Instant answers

No index, no schema definition, no rehydration

Data analytics at any time with Dynatrace Query Language

Cost efficient storage

No need to manage cold, warm or hot storage tiers

Blazing fast query performance

Massive parallel processing is up to 100 times faster

~1TB in ~1s @ 1000 cores

LOG MANAGEMENT AND ANALYTICS

powered by Grail

LAUNCHED in OCT 2022

Logs to Metrics

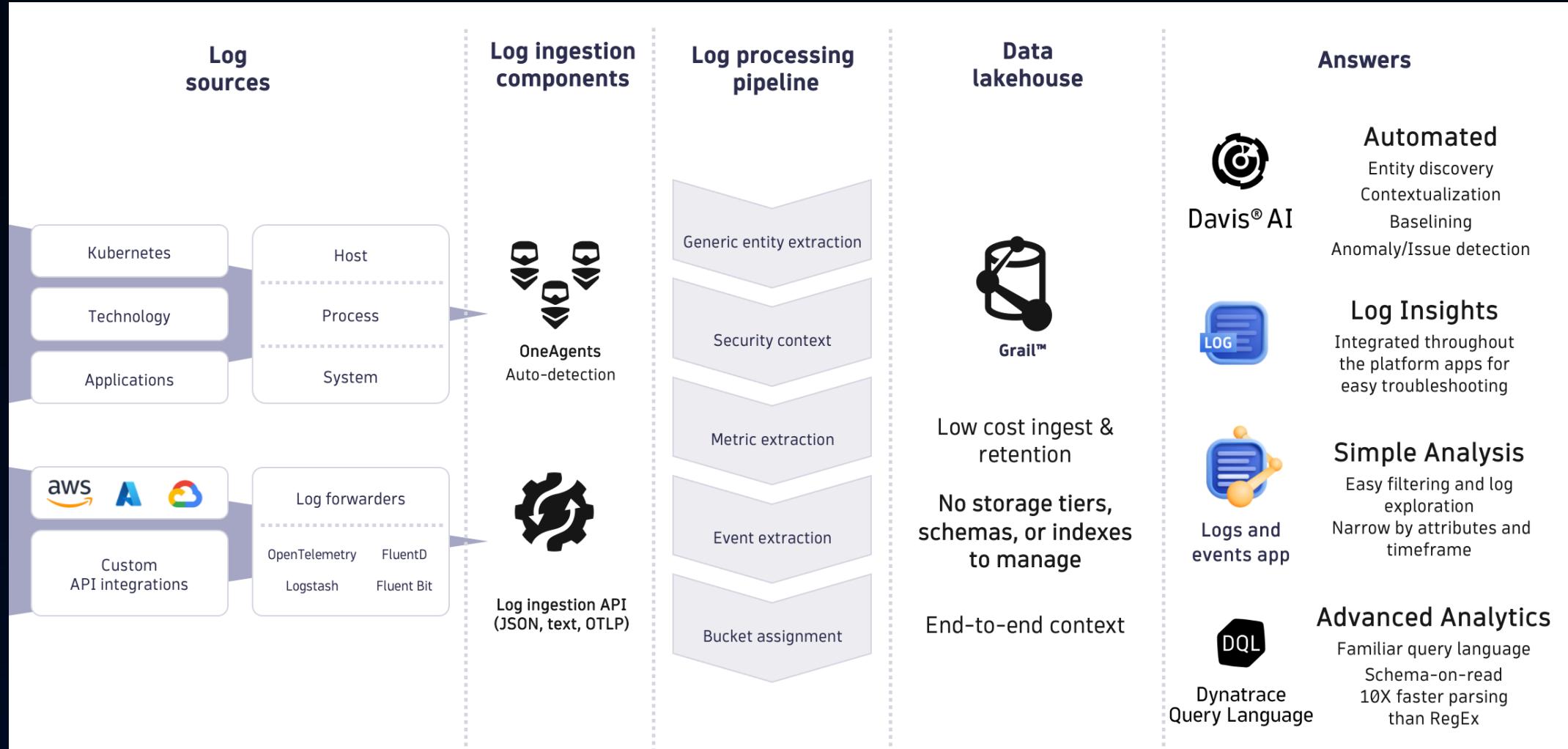
Troubleshooting

Application Optimization

Audit and Forensics



LOG MANAGEMENT & ANALYTICS ARCHITECTURE



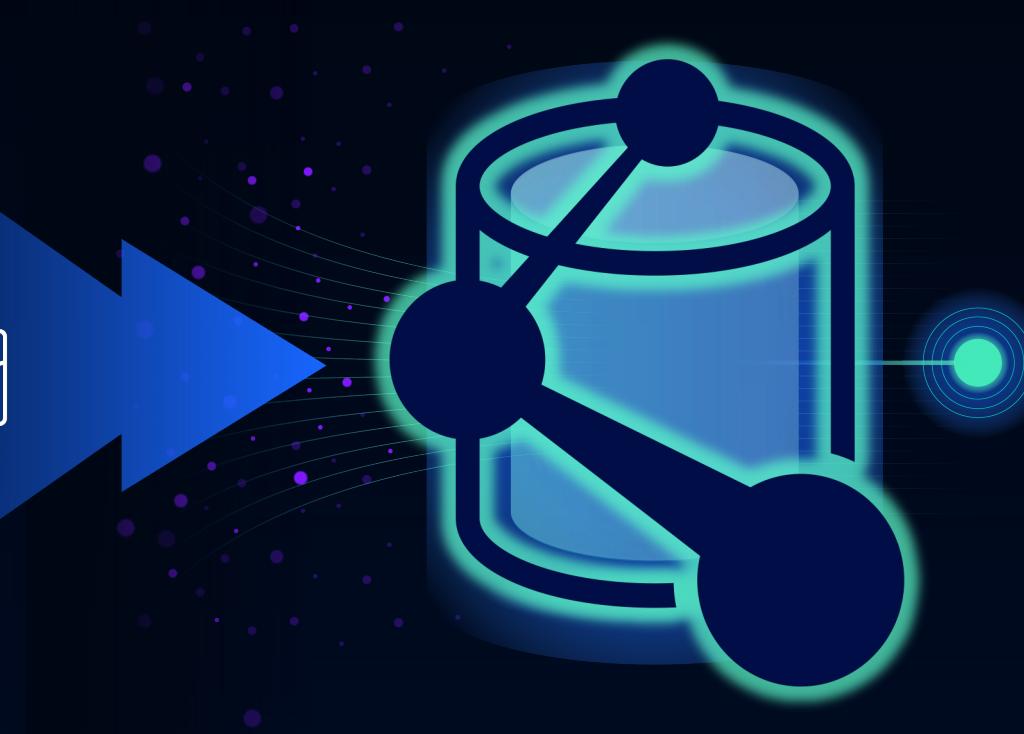
BUSINESS EVENTS

powered by Grail

LAUNCHED in NOV 2022

Extract business data
without code-changes

Lossless business data
for business dashboards



PERFORM

METRICS

powered by Grail

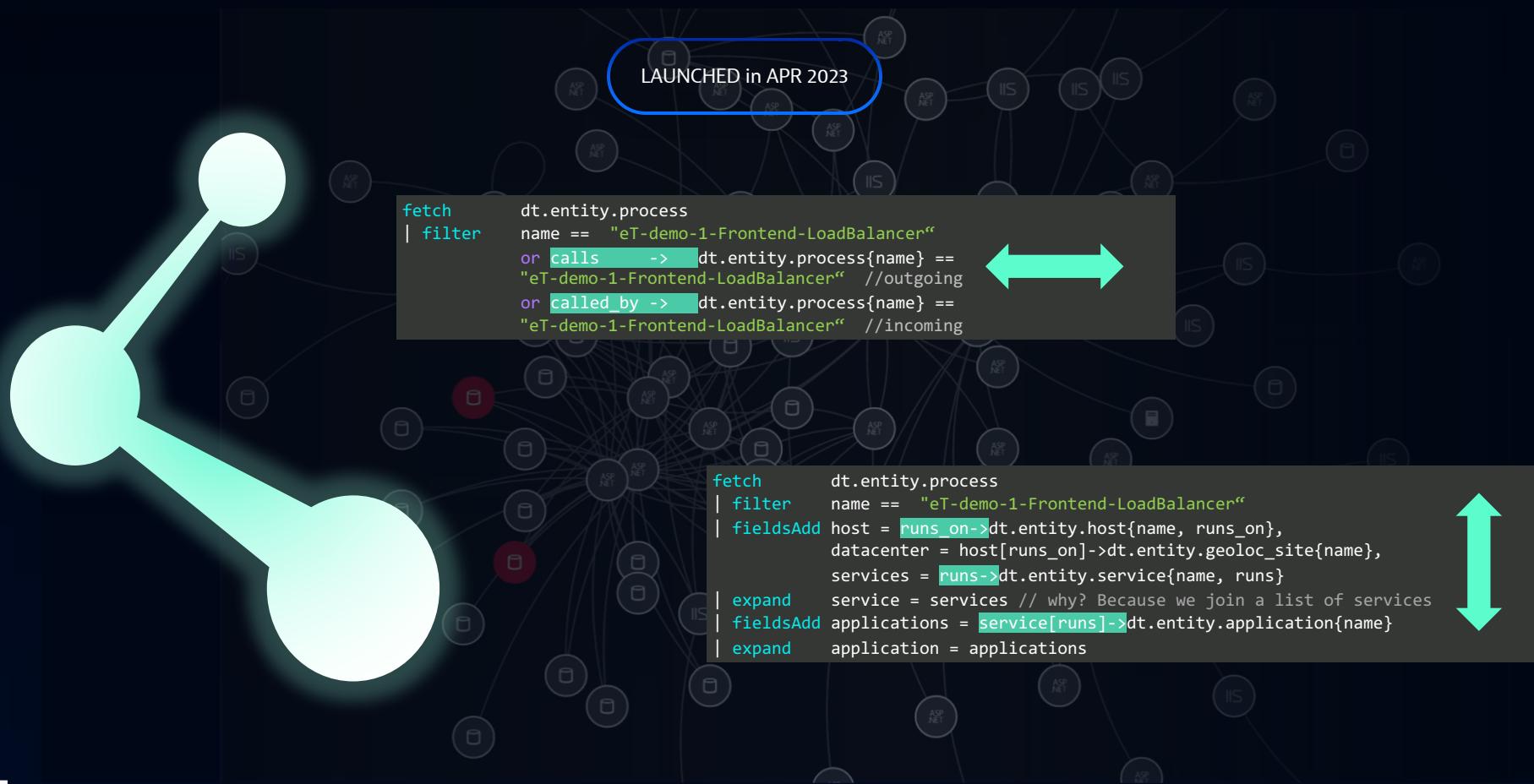
LAUNCHED in APR 2023



PERFORM

SMARTSCAPE FOR DQL

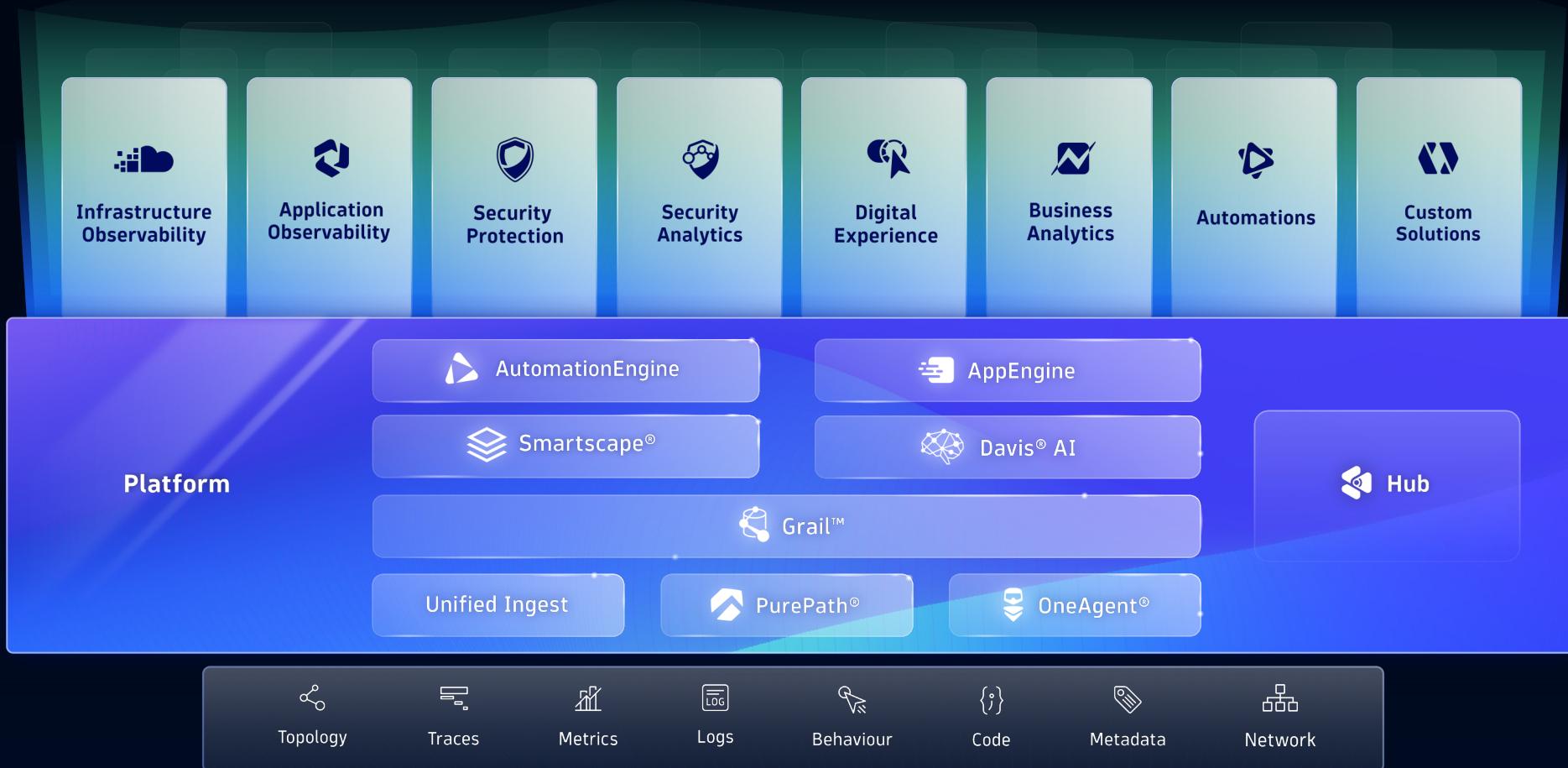
Gain the same power as Davis for
your own custom queries





Analytics and Automation for Unified Observability and Security

CLOUD DONE RIGHT.



HOW DO WE ACCESS ALL OF THIS DATA?



DYNATRACE QUERY LANGUAGE (DQL)

```
fetch      logs, from:now()-20m  
| filter   endswith(log.source, '/media/datasource/server-data')  
           and dt.host_group.id == "cluster_deve2e"  
| parse    content, "timestamp('yyyy-MM-dd HH:mm:ss'):ts"  
           1d json:settings  
           ipaddr:client_ip //IPv4/6"  
  
| fields   ts,  
           type = settings[eventType],  
           tenant = settings[tenantId],  
           user = settings[userId],  
           change = settings[jsonPatch]  
| filter   in(type, ["CREATE", "UPDATE", "DELETE"]) and type != "UNKNOWN"  
| summarize creates = countIf(type=="CREATE"),  
             by:{tenant, user}  
| fieldsAdd changes_per_min = (upd + del)/20  
| sort     changes_per_min desc
```

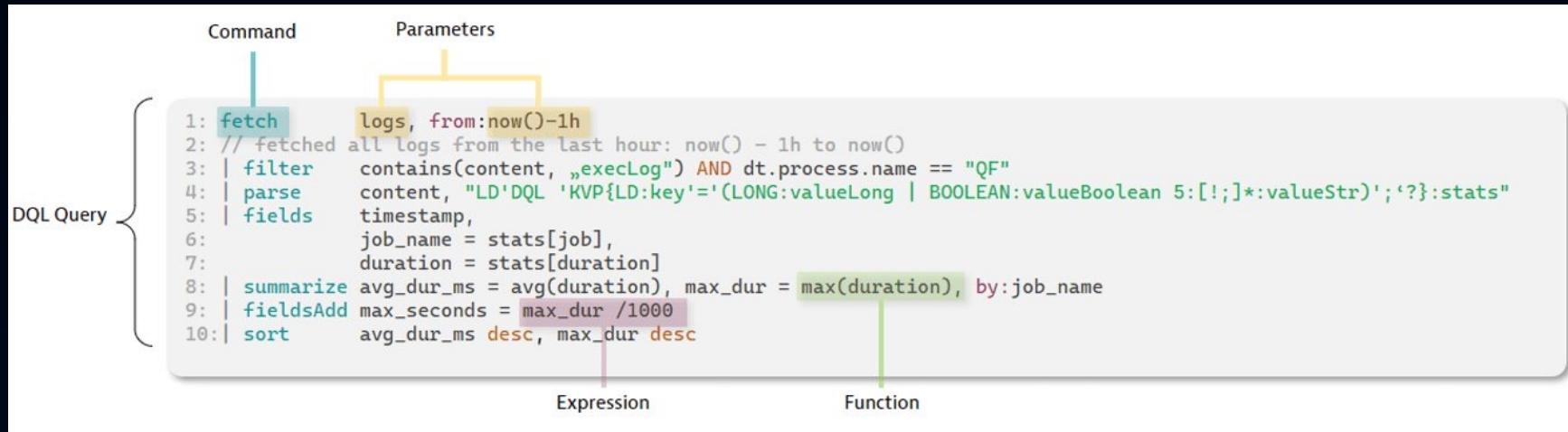


Purpose-built
for observability and security

Powerful
for even advanced use-cases,
parsing built-in

Effortless
migration from e.g. Splunk

WHAT IS DQL?



WHERE DO I USE DQL?

A series of thin, white, wavy lines that form a decorative pattern at the bottom of the slide. They start from the left edge, curve upwards and outwards towards the right, creating a sense of motion and depth against the blue gradient background.



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 [DQL](#)

Switch to Simple Mode

```
1 fetch logs //, scanLimitGBytes: 500, samplingRatio: 1000
2 | sort timestamp desc
```

[Run query](#)

Search results | Execution time: 1 s | Scanned data: 2.64 GB | Visualization type: [Table](#) [Single value](#) [Bar](#)

[Actions](#)

timestamp	status	content	loglevel	log.source	dt.process.name	trace_id	span_id	event.type	dt.entity.container.gro	dt.entity.container.gro	dt.e
2023-03-09 17:39:43	NONE	Mar 9 22:39:43 devkrzysztof multipathd[710]: sda: add missing path	NONE	/var/log/syslog	Linux System			LOG			
2023-03-09 17:39:43	NONE	Mar 9 22:39:43 devkrzysztof multipathd[710]: sda: failed to get udev uid: Invalid argument	NONE	/var/log/syslog	Linux System			LOG			
2023-03-09 17:39:43	NONE	Mar 9 22:39:43 devkrzysztof multipathd[710]: sda: failed to get sysfs uid: Invalid argument	NONE	/var/log/syslog	Linux System			LOG			
2023-03-09 17:39:43	NONE	Mar 9 22:39:43 devkrzysztof multipathd[710]: sda: failed to get sgio uid: No such file or directory	NONE	/var/log/syslog	Linux System			LOG			
2023-03-09 17:39:43	NONE	Mar 9 22:39:43 ip-10-1-92-64 kernel: [8494213.610217] [UFW AUDIT] IN= OUT=ens5 SRC=10.1.92.64 DST=52.4.169.68 LEN=508 TOS=0x00 PREC=0x00 TTL=64 ID=14678 DF PROTO=TCP SPT=43912 DPT=443 W...	NONE	/var/log/syslog	Linux System			LOG			
2023-03-09 17:39:43	NONE	Mar 9 22:39:43 ip-10-1-92-64 kernel: [8494213.611180] [UFW AUDIT] IN=ens5 OUT= MAC=0e:16:b3:b8:e3:4f:0e:06:f6:c1:0c:7c:08:00 SRC=52.4.169.68 DST=10.1.92.64 LEN=52 TOS=0x00 PREC=0x00 TTL=...	NONE	/var/log/syslog	Linux System			LOG			
2023-03-09 17:39:43	NONE	Mar 9 22:39:43 ip-10-1-92-64 kernel: [8494213.612109] [UFW AUDIT] IN=ens5 OUT= MAC=0e:16:b3:b8:e3:4f:0e:06:f6:c1:0c:7c:08:00 SRC=52.4.169.68 DST=10.1.92.64 LEN=353 TOS=0x00 PREC=0x00 TTL=...	NONE	/var/log/syslog	Linux System			LOG			
2023-03-09 17:39:43	NONE	Mar 9 22:39:43 ip-10-1-92-64 kernel: [8494213.612130] [UFW AUDIT] IN=ens5 OUT= MAC=0e:16:b3:b8:e3:4f:0e:06:f6:c1:0c:7c:08:00 SRC=52.4.169.68 DST=10.1.92.64 LEN=535 TOS=0x00 PREC=0x00 TTL=...	NONE	/var/log/syslog	Linux System			LOG			
2023-03-09 17:39:43	NONE	Mar 9 22:39:43 ip-10-1-92-64 kernel: [8494213.612335] [UFW AUDIT] IN=OUT=ens5 SRC=10.1.92.64 DST=52.4.169.68 LEN=523 TOS=0x00 PREC=0x00 TTL=64 ID=14680 DF PROTO=TCP SPT=43912 DPT=443 WI...	NONE	/var/log/syslog	Linux System			LOG			
2023-03-09 17:39:43	NONE	Mar 9 22:39:43 ip-10-1-92-64 kernel: [8494213.656787] [UFW AUDIT] IN=OUT=ens5 SRC=10.1.92.64 DST=52.4.169.68 LEN=523 TOS=0x00 PREC=0x00 TTL=64 ID=14679 DF PROTO=TCP SPT=43912 DPT=443 WI...	NONE	/var/log/syslog	Linux System			LOG			

Dashboards + New dashboard

Recent

Getting started

Saved 39 days ago

Share

Preview

+ Last 2 hours C

View all dashboards

Welcome to Dashboards

In this app you can:

- Query, visualize, and observe all your data stored in Grail
- Write custom JavaScript in Dynatrace functions to fetch external data
- Annotate all your visualizations with markdown to enrich them with context
- Add variables to filter your tiles and to make your dashboard dynamic

Try our quick guided tour

In our quick guided tour you'll learn how to get your first full functional dashboard up and running. It's a great place to start and there are no long explanations or tutorials.

Let's start with

1. Add your first visualization.

...

1. Add your first visualization

Steps

1. Select +
2. Select Add data
3. Other options include:
 - Data: to visualize your data stored in Grail using DQL.
 - Code: to visualize your external data.
 - Variable: we'll explore variables later on.
 - Annotation: to add context to your visualizations.
4. Select Run query and try out some different options.
5. Pin it to the dashboard with Add data and you're up and running.

Untitled dashboard Saved just now

Add data

```
1 fetch logs  
2 | summarize loglines = count(), by:{`1m interval` = bin(timestamp, 1m), status}
```

Run query

Run query to view results or learn more about the Dynatrace Query Language by clicking the link above.

Results

Result 1

Result 2

Result 3

NOTEBOOKS

Petabyte-scale data exploration & analytics for real-time insights



Explore

Know what Data is available and understand query options



Analyze

Powerful query language and interactive components allow customer analytics



Create data-driven documents

Persist results and enrich them with markdown



Collaborate

Share your created Analysis, Reports and Documentation with your Team

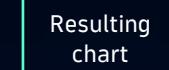
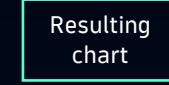


Extend

Functions allow the integration of external data sources and the implementation of custom analytics



PERFORM



The screenshot shows the Dynatrace Notebook interface with several sections:

- My Analytic Notebook**: A section showing a DQL query:

```
1 fetch logs, from:now()-90m  
2 | filter loglevel == "WARN" or loglevel == "ERROR"  
3 | summarize count = count(), by:{bin(timestamp, 1m), loglevel}  
4 | sort count desc
```

It also displays 162 records fetched at 8/31/2022, 1:19:50 PM, 51.5 kB. The "Charts" tab is selected.
- Chart library**: A histogram showing log level counts over time, with values ranging from 0 to 15.0 and time from 12:00 PM to 1:00 PM.
- JavaScript Function**: A code block showing a function to export weather data:

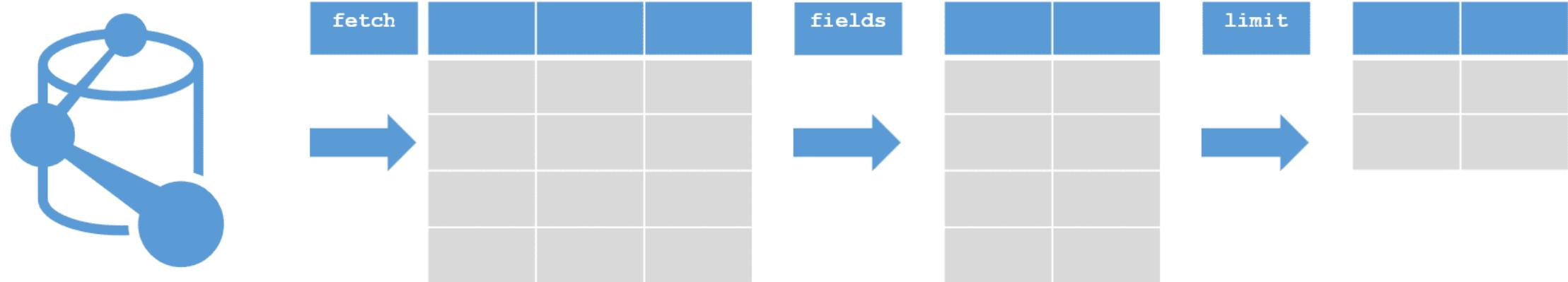
```
1 export default async () => {  
2   const json = await fetch('https://api.openweathermap.org/data/  
3   return json.main.temp;  
4 };
```

The "Charts" tab is selected.
- Resulting chart**: A large chart showing the current temperature as 23.52°C.
- Comment**: A section containing a summary and a DQL query:

```
# Summary  
This is what we've found using the following query  
```dql  
fetch logs
| filter dt.entity.host == "HOST-E73A576280286480"
| summarize count = count(), by:{`1m interval` = bin(timestamp, 1m)}
```
```

DQL Query Structure

Chaining Commands with the Pipe Operator



Logs and events



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 [DQL](#)

Switch to Simple Mode

```

1 fetch logs, from:now()-1h
2 // fetched all Logs from the last hour: now() - 1h to now()
3 | filter endsWith(log.source, "#.log")
4 | parse content, "LD IPADDR:ip ':' LONG:payload SPACE LD 'HTTP_STATUS' SPACE INT:http_status LD (EOL| EOS)"
5 | summarize total_payload = sum(payload),
6     failedRequests = countIf(http_status >= 400),
7     successfulRequests = countIf(http_status <= 400),
8     by:{ip, host.name}
9 | fieldsAdd total_payload_MB = total_payload/1000000
10 | fields ip, host.name, failedRequests, successfulRequests, total_payload_MB
11 | sort failedRequests desc

```

[Run query](#)

Search results

Visualization type: [Table](#) [Single value](#) [Bar](#)[Actions](#)

| ip | host.name | failedRequests | successfulRequests | total_payload_MB |
|------|---|----------------|--------------------|------------------|
| null | CF1-tomcatjms | 0 | 0 | 0 |
| null | CF2-tomcatjms | 0 | 0 | 0 |
| null | CF3-tomcatjms | 0 | 0 | 0 |
| null | CF4-tomcatjms | 0 | 0 | 0 |
| null | easytravel-demo1 | 0 | 0 | 0 |
| null | easytraveldemo-frontend1.wudrwjnln1ujodhhirvi2eb
xc.ax.internal.cloudapp.net | 0 | 0 | 0 |
| null | gke-alfa-cos-c9cc79-oalp.c.dynatrace-demoability.i
nternal | 0 | 0 | 0 |
| null | gke-alfa-cos-c9cc79-oz0j.c.dynatrace-demoability.in
ternal | 0 | 0 | 0 |

Logs and events



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 [DQL](#)

 Switch to Simple Mode

```

1 fetch      logs, from:now()-1h
2 // fetched all Logs from the last hour: now() - 1h to now()
3 | filter    endsWith(log.source, "#.log")
4 | parse     content, "LD IPADDR:ip ':' LONG:payload SPACE LD 'HTTP_STATUS' SPACE INT:http_status LD (EOL| EOS)"
5 | summarize total_payload = sum(payload),
6     failedRequests = countIf(http_status >= 400),
7     successfulRequests = countIf(http_status <= 400),
8     by:{ip, host.name}
9 | fieldsAdd total_payload_MB = total_payload/1000000
10 | fields   ip, host.name, failedRequests, successfulRequests, total_payload_MB
11 | sort    failedRequests desc

```

▶ Run query

Search results

Visualization type:

Table

Single value

Bar

Actions

| ip | host.name | failedRequests | successfulRequests | total_payload_MB |
|------|---|----------------|--------------------|------------------|
| null | CF1-tomcatjms | 0 | 0 | 0 |
| null | CF2-tomcatjms | 0 | 0 | 0 |
| null | CF3-tomcatjms | 0 | 0 | 0 |
| null | CF4-tomcatjms | 0 | 0 | 0 |
| null | easytravel-demo01 | 0 | 0 | 0 |
| null | easytraveldemo-frontend1.wudrwjnln1ujodhhirvi2eb
xc.ax.internal.cloudapp.net | 0 | 0 | 0 |
| null | gke-alfa-cos-c9cc79-oalp.c.dynatrace-demoability.i
nternal | 0 | 0 | 0 |
| null | gke-alfa-cos-c9cc79-oz0j.c.dynatrace-demoability.in
ternal | 0 | 0 | 0 |

Logs and events



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 [DQL](#)

 Switch to Simple Mode

```

1 fetch      logs, from:now()-1h
2 // fetched all Logs from the last hour: now() - 1h to now()
3 | filter    endsWith(log.source, "#.log")
4 | parse     content, "LD IPADDR:ip ':' LONG:payload SPACE LD 'HTTP_STATUS' SPACE INT:http_status LD (EOL| EOS)"
5 | summarize total_payload = sum(payload),
6     failedRequests = countIf(http_status >= 400),
7     successfulRequests = countIf(http_status <= 400),
8     by:{ip, host.name}
9 | fieldsAdd total_payload_MB = total_payload/1000000
10 | fields   ip, host.name, failedRequests, successfulRequests, total_payload_MB
11 | sort    failedRequests desc

```

▶ Run query

Search results

Visualization type: Table Single value BarActions

| ip | host.name | failedRequests | successfulRequests | total_payload_MB |
|------|---|----------------|--------------------|------------------|
| null | CF1-tomcatjms | 0 | 0 | 0 |
| null | CF2-tomcatjms | 0 | 0 | 0 |
| null | CF3-tomcatjms | 0 | 0 | 0 |
| null | CF4-tomcatjms | 0 | 0 | 0 |
| null | easytravel-demo01 | 0 | 0 | 0 |
| null | easytraveldemo-frontend1.wudrwjnln1ujodhhirvi2eb
xc.ax.internal.cloudapp.net | 0 | 0 | 0 |
| null | gke-alfa-cos-c9cc79-oalp.c.dynatrace-demoability.i
nternal | 0 | 0 | 0 |
| null | gke-alfa-cos-c9cc79-oz0j.c.dynatrace-demoability.in
ternal | 0 | 0 | 0 |



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 [DQL](#)

Switch to Simple Mode

```
1 fetch logs, from:now()-1h
2 // fetched all Logs from the last hour: now() - 1h to now()
3 | filter endsWith(log.source, "#.log")
4 | parse content, "LD IPADDR:ip ':' LONG:payload SPACE LD 'HTTP_STATUS' SPACE INT:http_status LD (EOL| EOS)"
5 | summarize total_payload = sum(payload),
6     failedRequests = countIf(http_status >= 400),
7     successfulRequests = countIf(http_status <= 400),
8     by:{ip, host.name}
9 | fieldsAdd total_payload_MB = total_payload/1000000
10 | fields ip, host.name, failedRequests, successfulRequests, total_payload_MB
11 | sort failedRequests desc
```

Run query

Search results | Visualization type: [Table](#) [Single value](#) [Bar](#)

| ip | host.name | failedRequests | successfulRequests | total_payload_MB | Actions |
|------|---|----------------|--------------------|------------------|---------|
| null | CF1-tomcatjms | 0 | 0 | 0 | |
| null | CF2-tomcatjms | 0 | 0 | 0 | |
| null | CF3-tomcatjms | 0 | 0 | 0 | |
| null | CF4-tomcatjms | 0 | 0 | 0 | |
| null | easytravel-demo1 | 0 | 0 | 0 | |
| null | easytraveldemo-frontend1.wudrwjnln1ujodhhirvi2eb
xc.ax.internal.cloudapp.net | 0 | 0 | 0 | |
| null | gke-alfa-cos-c9cc79-oalp.c.dynatrace-demoability.i
nternal | 0 | 0 | 0 | |
| null | gke-alfa-cos-c9cc79-oz0j.c.dynatrace-demoability.in
ternal | 0 | 0 | 0 | |



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()-1h
2 // fetched all logs from the last hour: now() - 1h to now()
3 | filter    endsWith(log.source, "#.log")
4 | parse     content, "LD IPADDR:ip ':' LONG:payload SPACE LD 'HTTP_STATUS' SPACE INT:http_status LD (EOL| EOS)"
5 | summarize total_payload = sum(payload),
6             failedRequests = countIf(http_status >= 400),
7             successfulRequests = countIf(http_status <= 400),
8             by:{ip, host.name}
9 | fieldsAdd total_payload_MB = total_payload/1000000
10 | fields    ip, host.name, failedRequests, successfulRequests, total_payload_MB
11 | sort      failedRequests desc
```

Run query





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()-1h
2 // fetched all logs from the last hour: now() - 1h to now()
3 | filter    endsWith(log.source, "#.log")
4 | parse     content, "LD IPADDR:ip ':' LONG:payload SPACE LD 'HTTP_STATUS' SPACE INT:http_status LD (EOL| EOS)"
5 | summarize total_payload = sum(payload),
6             failedRequests = countIf(http_status >= 400),
7             successfulRequests = countIf(http_status <= 400),
8             by:{ip, host.name}
9 | fieldsAdd total_payload_MB = total_payload/1000000
10 | fields    ip, host.name, failedRequests, successfulRequests, total_payload_MB
11 | sort      failedRequests desc
```

Run query





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()-1h
2 // fetched all logs from the last hour: now() - 1h to now()
3 | filter    endsWith(log.source, "#.log")
4 | parse     content, "LD IPADDR:ip ':' LONG:payload SPACE LD 'HTTP_STATUS' SPACE INT:http_status LD (EOL| EOS)"
5 | summarize total_payload = sum(payload),
6             failedRequests = countIf(http_status >= 400),
7             successfulRequests = countIf(http_status <= 400),
8             by:{ip, host.name}
9 | fieldsAdd total_payload_MB = total_payload/1000000
10 | fields    ip, host.name, failedRequests, successfulRequests, total_payload_MB
11 | sort      failedRequests desc
```

Run query





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()-1h
2 // fetched all logs from the last hour: now() - 1h to now()
3 | filter    endsWith(log.source, "#.log")
4 | parse     content, "LD IPADDR:ip ':' LONG:payload SPACE LD 'HTTP_STATUS' SPACE INT:http_status LD (EOL| EOS)"
5 | summarize total_payload = sum(payload),
6             failedRequests = countIf(http_status >= 400),
7             successfulRequests = countIf(http_status <= 400),
8             by:{ip, host.name}
9 | fieldsAdd total_payload_MB = total_payload/1000000
10 | fields    ip, host.name, failedRequests, successfulRequests, total_payload_MB
11 | sort      failedRequests desc
```

Run query





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()-1h
2 // fetched all logs from the last hour: now() - 1h to now()
3 | filter    endsWith(log.source, "#.log")
4 | parse     content, "LD IPADDR:ip ':' LONG:payload SPACE LD 'HTTP_STATUS' SPACE INT:http_status LD (EOL| EOS)"
5 | summarize total_payload = sum(payload),
6             failedRequests = countIf(http_status >= 400),
7             successfulRequests = countIf(http_status <= 400),
8             by:{ip, host.name}
9 | fieldsAdd total_payload_MB = total_payload/1000000
10 | fields    ip, host.name, failedRequests, successfulRequests, total_payload_MB
11 | sort      failedRequests desc
```

Run query



HANDS ON LABS



LAB ENVIRONMENT



- Dynatrace is already capturing data from 2-3 apps on a K8s cluster
 - Metrics, Logs, Traces, Biz Events, AppSec, etc.
- Access the environment via Dynatrace University
 - Navigate to Dashboard
 - Upcoming Events – Intro to DQL
 - Environments Tab
 - Contains your Dynatrace environment link
- Exercises are provided as read-only Notebooks in the Dynatrace tenant. Answer key Notebooks will be provided at the end of each sections.
- Recommendation: Close all other browser windows/tabs other than Dynatrace University and the Dynatrace tenant.

LAB ENVIRONMENT LOGIN

- Access the environment via Dynatrace University
 - Navigate to Dashboard
 - Upcoming Events – Intro to DQL
 - Environments Tab
 - Contains your Dynatrace environment
- From the environments tab in Dynatrace University, click VIEW ENVIRONMENT and verify you can login into the Dynatrace tenant
- Click NOTEBOOKS and open Logs Part 1



PERFORM

The image shows two screenshots of the Dynatrace interface. The top screenshot is titled 'Lab environments' and displays a 'Dynatrace Environment' section with a button labeled 'View environment'. Below this is a 'Good morning' greeting and a section titled 'Here are some tips to help you explore the latest Dynatrace' which includes links to 'Understand data with Smartscape', 'Explore data with Notebooks', 'Observe data with all-new Dashboards', 'Automate and Integrate', and 'Get more data into Dynatrace'. It also features a 'Smartscape' visualization and a 'See how everything works together' callout. The bottom screenshot shows the main Dynatrace dashboard with a navigation bar at the top and sections for 'Notebooks', 'Dashboards', 'Dashboards Classic', 'Data explorer', 'Metrics', 'Logs & Events Classic', 'Problems', and 'Smartscape topology'.

BIZ EVENTS



MISALIGNMENT BETWEEN BUSINESS AND IT TEAMS

Challenges

Web analytics solutions

- Narrow user journey and conversion focus
- Business metrics are inaccurate

Business intelligence tools

- Lack real-time insights into business KPIs
- Lack IT context, limiting collaboration

Observability solutions

- Lack precision
- Have limited access to business data
- Are complex to configure and maintain, often requiring custom code

Implications

Business leaders rely on data that is stale, imprecise, and not connected to IT



WHAT IF YOU COULD EASILY ACCESS PRECISE, CONTEXTUAL BUSINESS DATA?

Precise and granular

- No sampling, no extrapolation
- No aggregation

Long term

- Multi-year retention for trending, compliance, ad-hoc analytics

Easy to access

- Extract data from application payloads – without custom code

Externally enriched

- Easy to ingest data from external business tools

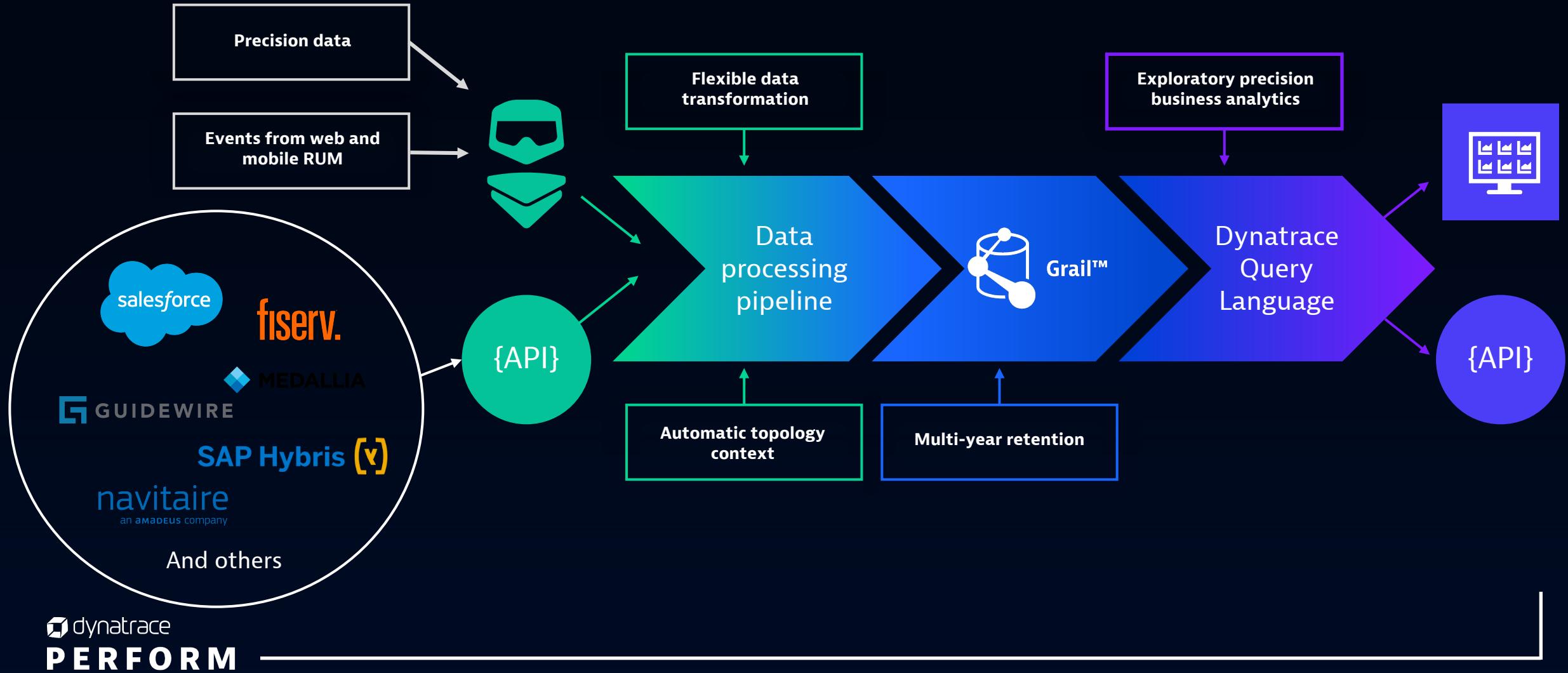
Automatic context

- Effective collaboration requires full-stack topology context

Unified storage

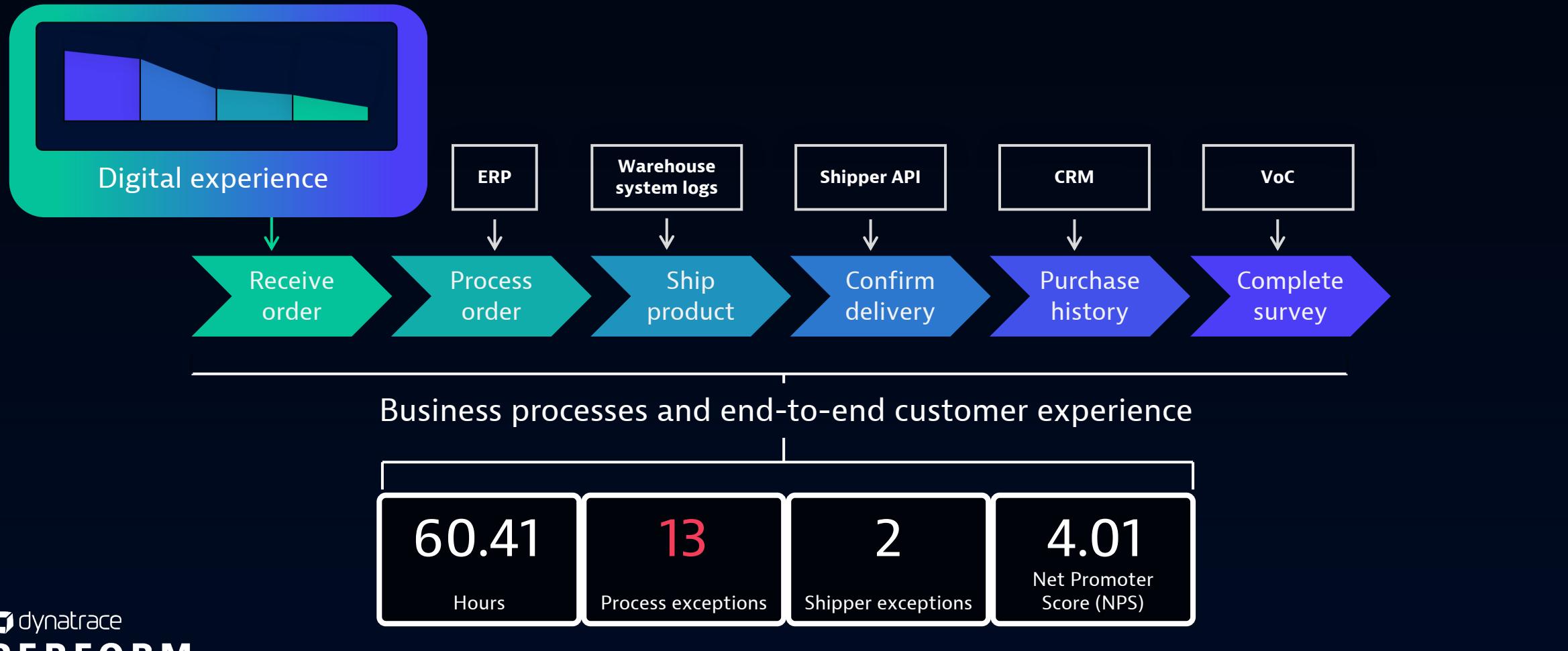
- Long-term and indexless
- No need to predict future questions

BUSINESS EVENTS: ARCHITECTURAL OVERVIEW



GAIN VISIBILITY INTO COMPLETE BUSINESS PROCESSES

Business events expand use cases beyond the digital conversion



POWER BUSINESS DASHBOARDS, EXPLORATORY ANALYTICS, AND AUTOMATION

Report business KPIs and explore all your business data interactively

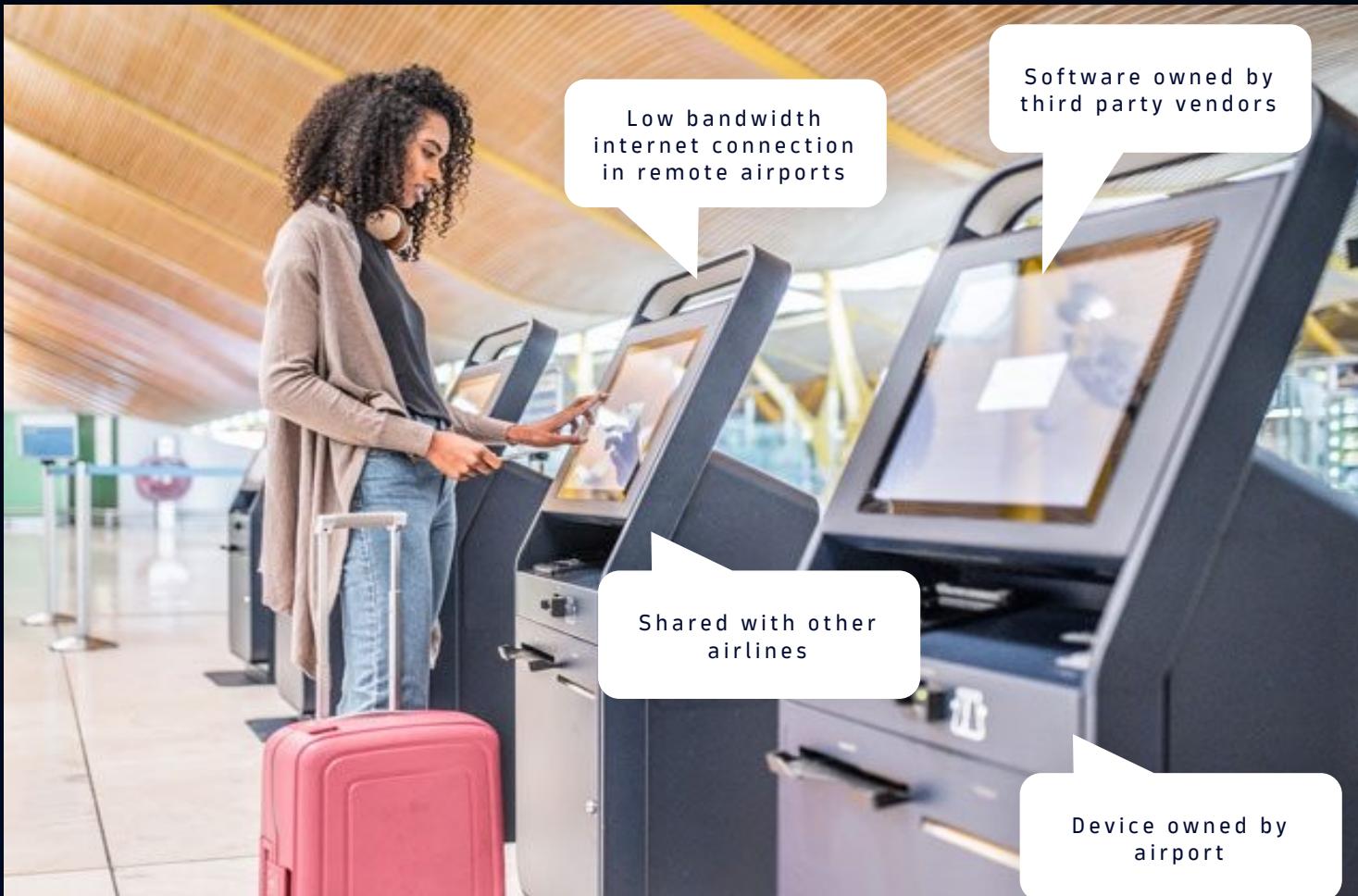
The image displays three distinct but interconnected interfaces:

- Business Dashboard (Left Panel):** Shows real-time KPIs like 'Orders Processed' (3597), 'Orders Sales' (\$35790.0), and 'Net Promoter Score (NPS)' (85.0). It includes a timeline chart for 'Orders Processed' from 11 AM to 12:30 PM, a bar chart for 'Orders by Geo' (West, Midwest, East, South), and a 'Customer Reviews Summary' bar chart.
- Order Processing and Warehouse Health (Middle Panel):** Monitors processing times: 'Order to Warehouse' (8.75 hrs), 'Warehouse to Shipment' (24.71 hrs), and 'Shipped to Delivered' (21.99 hrs). It features a 'Order to Delivery (hrs)' line chart and a 'Warehouse Failure' bar chart.
- Notebooks (Right Panel):** A Jupyter-style notebook titled 'Trading Analysis' with the following code:

```
1 fetch bizevents
2 | filter event.type == "com.easystrade.buy-assets"
3 | fieldsAdd TradingVolume = amount*price
4 | summarize sum(TradingVolume), alias: Trading_Volume, by: {bin(timestamp, 1h)}
```

A histogram visualizes the 'Trading_Volume' over time from May 31 to June 3, 2023.

HOW TO EFFICIENTLY MANAGE AND OPTIMIZE AIRLINE KIOSKS WITHOUT FULL CONTROL?



- Kiosks owned by third party preventing instrumentation
- Transferring log files from thousands of kiosks is error-prone and not real time
- Staffing kiosk areas to provide assistance is expensive

MAPPING THE ENTIRE KIOSK EXPERIENCE USING A SINGLE ONEAGENT

- Initial visibility into every kiosk action along the funnel:
 - Success/failure rates by record lookup method
 - Filterable by airport and down to individual kiosks
- Expanded to key business and marketing metrics:
 - City-pair statistics
 - Funnel comparisons between checked-in statuses
 - Statistics between native kiosks and "common use" kiosks that service multiple airlines

100%
OF EXPERIENCE
mapped out across self-service platform

COST SAVINGS

by reducing support and staffing for kiosks

INCREASED COMPLETION RATES
for passengers through self check-in

INCREASED REVENUE

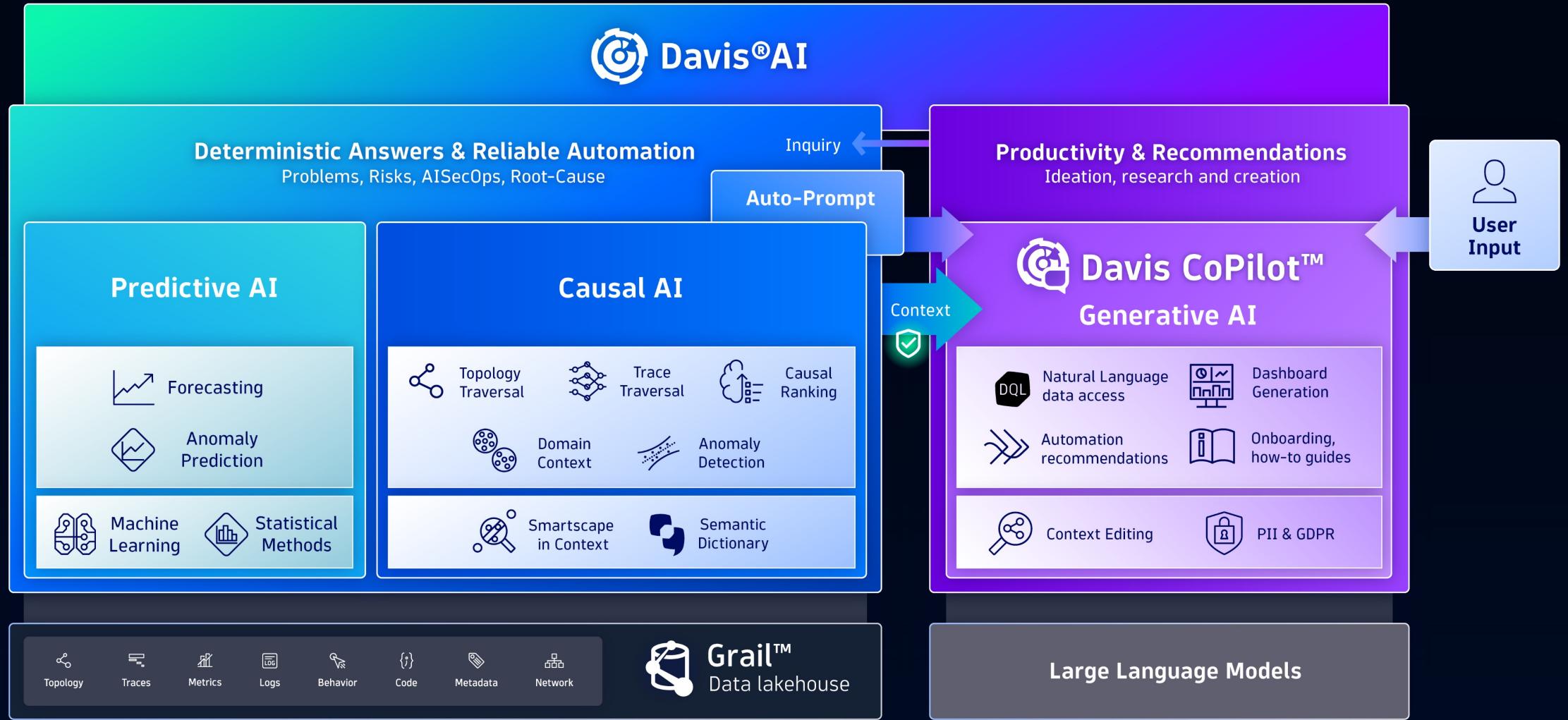
associated with flight add-ons

GENERATIVE AI



Natural Language to DQL and Conversational Answers

HYPERMODAL AI FOR UNIFIED OBSERVABILITY AND SECURITY



NATURAL LANGUAGE QUERIES

The screenshot shows the Dynatrace Perform interface. On the left is a sidebar with various notebooks listed under 'Recent' and 'All notebooks'. The main area displays a chart titled 'Top list for the average of metric dt.kubernetes.note.requests_cpu split by nodes'. The chart shows CPU usage for several K8s nodes. A modal window in the foreground is titled 'Davis CoPilot™' and asks 'How can I help you with a query?'. It contains the text 'Show the top 5 K8's nodes by CPU.' and a 'Create' button.

Recent

Notebooks

+ New notebook

CPU analysis

Payment proc. anomaly

Opportunity analytics

Revenue allocation

Black Friday report 2021

CVE-2022-22965 report

log4shell: remediation

Track emerging threats

Troubleshoot SSO

Investigate disk usage

Pattern usage

All notebooks

»

?

CPU analysis

Saved 6 minutes ago

Share

Rerun sections

Top list for the average of metric dt.kubernetes.note.requests_cpu split by nodes

```
1 timeseries avg_cpu_req = avg(dt.kubernetes.node.requests_cpu), by:{k8s.node.name}
2 | fieldsAdd total = arrayAvg(avg_cpu_req)
3 | fields k8s.node.name, total
4 | sort total desc
5 | limit 5
```

5 records Executed at: 13/01/2023, 05:33:11 Timeframe: 05:33:11 - 07:33:11 ⓘ

k8snode name

| k8snode name | avg_cpu_req |
|--|-------------|
| unguard | ~450 |
| ip-10-178-54-32.ec2.internal | ~350 |
| car-deb10-oneagent-01 | ~300 |
| deb-10-k3s-oi-01 | ~250 |
| gke-openfeature-argocd-d-default-pool-a2bc2f01-h41 | ~200 |

Is this helpful? ⓘ ✓ Done ⚙ Adjust

Davis CoPilot™

How can I help you with a query?

Show the top 5 K8's nodes by CPU.

Create

CONVERSATIONAL RECOMMENDER

New conversation X



Davis CoPilot™

Get more done with your new productivity assistant.

Infrastructure Observability

Are there any resources nearing capacity?

Application Observability

Are there any current system outages or incidents?

Digital Experience

How are page load times affecting user experience?

Automations

Did the latest deployment introduce any performance issues?

Application Security

Show me the latest security vulnerabilities affecting our stack.

Business Analytics

What's the correlation between system downtime and sales?

Explore more prompts ↗

Hey, how is the Search API Service performance this week? ↗

Davis CoPilot™ is a generative AI that may produce inaccurate answers.

Search API Service performance + New conversation X

Here's an overview of the Search API Service performance for the last week. Here are some key performance metrics and insights:

1. Response Times: On average, the response time for the Search API Service was 450 milliseconds this week. However, there were occasional spikes that reached up to 1200 milliseconds.
2. Error Rates: The error rate remained relatively low, with only 0.4% of all requests resulting in errors. These errors were mostly related to occasional network timeouts.
3. Throughput: The Search API Service experienced steady growth in user traffic. The average throughput increased by 5.8% compared to the previous week, indicating a growing user base.
4. Anomalies: 1 anomaly was observed in the server response time on **Tuesday, July 18** last week. Further investigation may be required to determine the cause and prevent future occurrences.

Was this answer helpful? ✓ ✗

4 sources How does the error rate compare to our SLO?

Over the last week, the Search API Service has maintained 99.6% uptime, exceeding the SLO target of 99.00% uptime.

Here's a chart indicating the Search API Service SLO over the last week.



Open in Service-level Objectives ↗ Open with ↗

Was this answer helpful? ✓ ✗

1 source Ask for help or insights ↗

Ask for help or insights ↗

Davis CoPilot™ is a generative AI that may produce inaccurate answers.

Talk to a Dynatrace expert ↗

WHAT ELSE IS POSSIBLE?

AUTOMATIC RCA

Root cause
Based on time correlation and analysis of all transactions that use these components, this issue has the following root cause

JourneyService
Web service

3 Response time degradation events
The current response time (~308.16 ms) exceeds the auto-detected baseline (~40.68 ms) by 652.59 %. Service method findLocations has a slowdow...

Events on:
Service JourneyService

Analyze code

Metric anomalies detected
Review the metrics which show anomalous behavior.

Visual resolution
Click to see how we figured this out.

Davis CoPilot™
How can I help you with Problems?
Summarize all my open problems and suggest appropriate remediations.
Send

AUTO-CODED WORKFLOWS

Davis CoPilot™
How can I help you?

Create a workflow to trigger the creation of a notification that will be directed to the correct team handling vulnerability analysis.

Create

get_enriched_security_pr...
Load Problem details and affected entities data.

get_problem_details
Load Problem details and affected entities data

escalate_critical_issues
Sending email via Dynatrace

get_existing_tickets
Search issue using JQL

New task
Select an action from the list

create_tickets
Automation action to be used to create issues on Jira

update_tickets
Jira: Comment on ticket

notify_responsible_teams
for each affected entity

Site Reliability Guardian
Site Reliability Guardian
Automation action to start a site reliability guardian validation

Slack Integration
Slack: Send message
Send a message to a Slack workspace

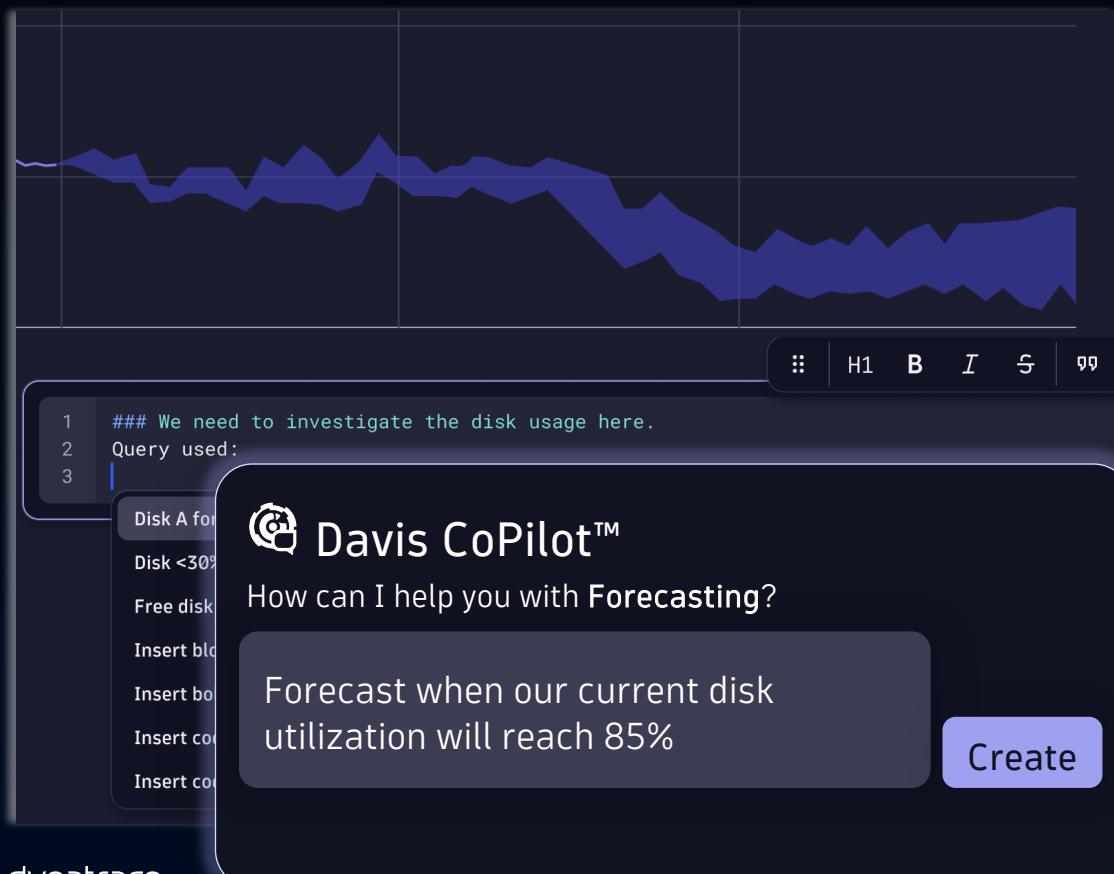
Security Problem Remediation
Get Enriched Security Problem
Load Problem details and affected entities data.

Demo Mail
Send email
Sending email via Dynatrace

Ansible Tower
Cancel job

WHAT ELSE IS POSSIBLE?

PREDICTIVE OPS



AUTO-GENERATED SRG

Objectives

Objective 1

Objective name*
Payment Service Workload

Description
Workload of the payment service

DQL query*

```
1 timeseries throughput = max(dt.service.request.count),  
2 filter: in(dt.entity.service, "SERVICE-F1AD53B6B0C761E5")  
3 | fields max = throughput
```

Run query

Preview your query

Run your query and set warn
The query result previews ho



Davis CoPilot™

What would you like to achieve?

I would like to create a guardian for my payment service running in production. Throughput should be stable with new releases /versions.

Send

WHAT ELSE IS POSSIBLE?

DASHBOARD CREATION

Product information

Davis CoPilot™

How can I help you with Dashboards?

Create a dashboard with my inventory information. Set a variable based on geolocation/region.

Create

Stock

| Product | Value 1 | Value 2 |
|-------------------|---------|---------|
| Apple | 100 | 30 |
| Samsung | 30 | 50 |
| OPPO | 130 | 120 |
| Huawei | 40 | 30 |
| Apple | 90 | 80 |
| Samsung | 60 | 50 |
| Microsoft Surface | 70 | 60 |
| Infinix | 100 | 90 |
| HP Pavilion | 90 | 80 |
| Impresso.Ua Di Gi | 70 | 60 |
| Royal Mirage | 50 | 40 |
| Fog Scent Xpresso | 60 | 50 |
| Al Munakh | 120 | 110 |
| Lord - Al-Rehab | 110 | 100 |
| L'Oreal Paris | 110 | 100 |
| Hemani Tea | 80 | 70 |
| Dermive | 90 | 80 |
| ROREC White Rice | 50 | 40 |
| Fair & Clear | 140 | 130 |
| Saaf & Khadas | 140 | 130 |
| Bake Pardon Big | 150 | 140 |
| Baking Food Items | 20 | 10 |
| fauji | 120 | 100 |
| Dry Rose | 50 | 40 |
| Boho Decor | 140 | 130 |
| Flying Wooden | 20 | 10 |
| LED Lights | 60 | 50 |
| luxury palace | 10 | 5 |
| Golden | 50 | 40 |

SECURITY ANALYTICS

Davis CoPilot™

How can I help you with Security Analytics ?

Based on session IDs of 172.31.24.11 in my access log, can you find authentication responses containing the same session IDs in my webapp log? What are the top auth responses for that session ID?

Send

```
1   fetch logs, from:-30d
2   | filter contains(log.source, "ludo-clinic-webapp.log")
3   | parse content, "'[ ' TIMESTAMP('dd/MMM/yyyy:HH:mm:ss.S') :event_time LD '
4   | limit 10000
5   | fields event_time, sessionIdApp, result_text, username, content
6   | filter contains(content, "CustomAuthenticationProvider")
7   AND NOT contains(content, "Starting findUsersByUsernameAndPassword") //
8   AND NOT contains(content, "retrieved matching list")
```

ROADMAP AND ADDITIONAL LEARNING RESOURCES



KEY REFERENCE LINKS



- Download the Notebooks so you have them as references!
- Learn DQL Interactive App (Good place to start for beginners):
https://<DT_TENANT_ID>.apps.dynatrace.com/ui/apps/dynatrace.learndql/
- DQL Course on Dynatrace University (More in-depth content):
<https://university.dynatrace.com/ondemand/course/39387?content=overview>
- Documentation:
<https://www.dynatrace.com/support/help/shortlink/dql-dynatrace-query-language-hub>

**One Community to find them all,
one Community to solve them.**

Join other observability experts
at community.dynatrace.com



FEEDBACK



THANK YOU



Enjoy Perform 2024!