



# Logging: Best Practices and Tools

Anderson Queiroz

---

Thursday 6th July 2023

# \$ whoami



Anderson Queiroz | Sr. Software Engineer

- Brazilian living in Germany for 5+ years
- 15+ coding
- ~5 years working with Go
- 1.5+ year at Elastic
- Platform | Ingest | Elastic Agent team
- Hobbies include:
  - Cooking and mixology (making cocktails)
  - Muay-Thai (Thai Boxing) / Exercise
  - 3D printing (resing)
  - Gaming

# What is logging?



# What is Logging

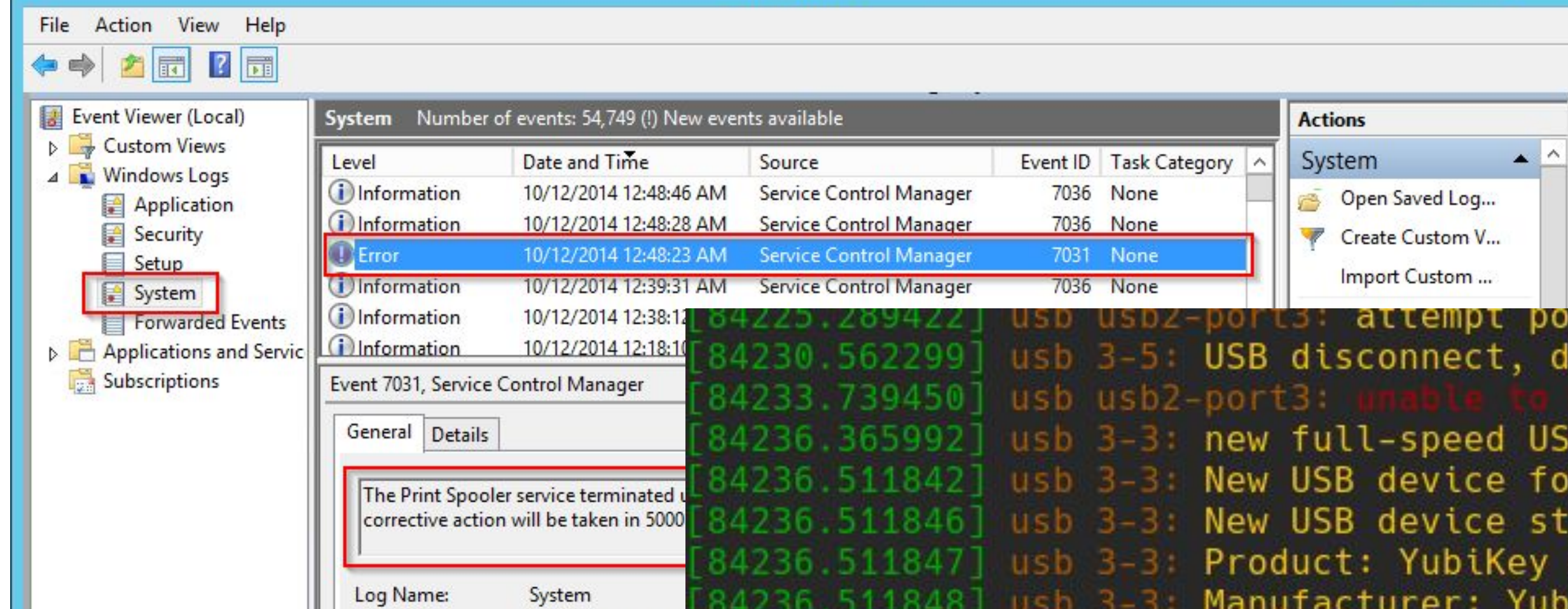
It's how a program talks to you, how it asks for help.

Simplicity put, it's information a program produces about what it's doing, how it's doing, data or metrics about itself.

It's one, if not, the primary source of information to understand how the software is running, if errors are happening and investigate issues.

More formally, logging is a computer system, application, or program systematically producing messages or records of events during its execution.





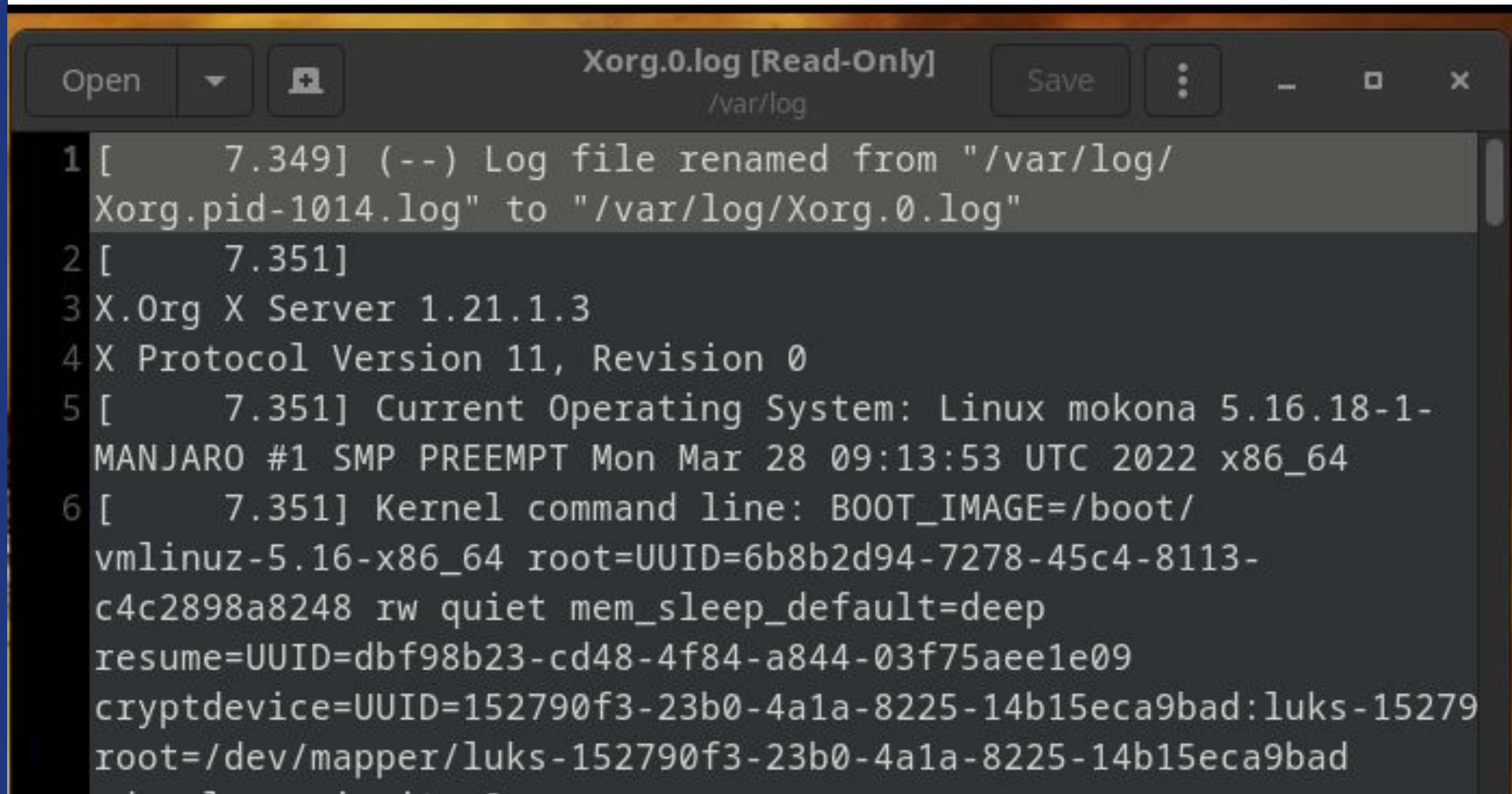
```
[84225.289422] usb usb2-port3: attempt power cycle
[84230.562299] usb 3-5: USB disconnect, device number 7
[84233.739450] usb usb2-port3: unable to enumerate USB device
[84236.365992] usb 3-3: new full-speed USB device number 8 using xhci_hcd
[84236.511842] usb 3-3: New USB device found, idVendor=1050, idProduct=0407, bcdDevice=00.00
[84236.511846] usb 3-3: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[84236.511847] usb 3-3: Product: YubiKey OTP+FIDO+CCID
[84236.511848] usb 3-3: Manufacturer: Yubico
[84236.515693] input: Yubico YubiKey OTP+FIDO+CCID as /devices/pci0000:00/0000:00:02:00/input/input1
[84236.569693] hid-generic 0003:1050:0407.0009: input,hidraw1: USB HID v1.10 Keyboard [Yubico YubiKey OTP+FIDO+CCID] on usb-0000:00:02:00
[84236.570483] hid-generic 0003:1050:0407.000A: hiddev96,hidraw2: USB HID v1.10 Device [Yubico YubiKey OTP+FIDO+CCID] on usb-0000:00:02:00
[84244.135917] usb 3-3: USB disconnect, device number 8
[84394.976146] usb 3-10: reset full-speed USB device number 2 using xhci_hcd
[84395.246113] usb 3-10: reset full-speed USB device number 2 using xhci_hcd
21 Jun, 2023 @ 09:34:58.208 update marker not present at /opt/Elastic/Agent/data
21 Jun, 2023 @ 09:34:58.159 Detected available inputs and outputs
21 Jun, 2023 @ 09:34:58.159 Capabilities file not found in /opt/Elastic/Agent/capabilities.yml
21 Jun, 2023 @ 09:34:58.159 Determined allowed capabilities
21 Jun, 2023 @ 09:34:58.144 Gathered system information
21 Jun, 2023 @ 09:34:58.005 APM instrumentation disabled
```



# What is Logging

A logfile is a file containing several log lines or entries from one or more systems.

Usually in text format, but, as any data, there are several ways to store it. For example, Journald logs are stored in a binary format.

A screenshot of a text editor window showing the contents of a log file named 'Xorg.0.log'. The window title is 'Xorg.0.log [Read-Only]' and the file path is '/var/log'. The log content includes a timestamped message about a log file rename, the X.Org X Server version (1.21.1.3), the X Protocol version (11, Revision 0), the current operating system (Linux mokona 5.16.18-1-MANJARO #1 SMP PREEMPT Mon Mar 28 09:13:53 UTC 2022 x86\_64), and the kernel command line with various boot parameters like BOOT\_IMAGE, root=UUID, and cryptdevice.

```
1 [ 7.349] (--) Log file renamed from "/var/log/Xorg.pid-1014.log" to "/var/log/Xorg.0.log"
2 [ 7.351]
3 X.Org X Server 1.21.1.3
4 X Protocol Version 11, Revision 0
5 [ 7.351] Current Operating System: Linux mokona 5.16.18-1-MANJARO #1 SMP PREEMPT Mon Mar 28 09:13:53 UTC 2022 x86_64
6 [ 7.351] Kernel command line: BOOT_IMAGE=/boot/vmlinuz-5.16-x86_64 root=UUID=6b8b2d94-7278-45c4-8113-c4c2898a8248 rw quiet mem_sleep_default=deep resume=UUID=dbf98b23-cd48-4f84-a844-03f75aee1e09 cryptdevice=UUID=152790f3-23b0-4a1a-8225-14b15eca9bad:luks-152790f3-23b0-4a1a-8225-14b15eca9bad
```

# What is in a log

# What is in a log



**Timestamp or temporal/order identifier**  
It's important to know the order of events, and when they happened.



**Message**  
The core of the log, what you want to inform about. A human readable sentence explaining what happened.



**Level**  
Also called severity is the importance or category of the log. The most common levels are *trace*, *debug*, *info*, *warn*, *error* and *fatal*.



**Extra information**  
Any other information that will add context to the message.



# Structured logging

# Structured logging



## **Logs are for humans to read**

First and foremost always keep in mind that logs are for humans to read and understand what is happening.



## **Logs need to be searched**

Usually we look for the right information among thousands to a plethora of logs. They need to be machine-readable so we can search them.



## **Logs need to be correlated**

We need to identify the log entries belonging to the same flow, process, user, context whatever we're are investigating.

# Structured logging

```
INFO[0000] A group of walrus emerges from the ocean    animal=walrus size=10
WARN[0000] The group's number increased tremendously!  number=122 omg=true
INFO[0000] A giant walrus appears!                    animal=walrus size=10
INFO[0000] Tremendously sized cow enters the ocean.   animal=walrus size=9
FATA[0000] The ice breaks!                            number=100 omg=true
exit status 1
```

Source: <https://camo.githubusercontent.com/369a631bb41ad67da0d3d95423a704319f488049d25eac55ee7295363a7afb81/687474703a2f2f692e696d6775722e636f6d2f505937714d77642e706e67>

```
5:41PM INF Starting listener listen=:8080 pid=37556
5:41PM DBG Access database=myapp host=localhost:4932 pid=37556
5:41PM INF Access method=GET path=/users pid=37556 resp_time=23
5:41PM INF Access method=POST path=/posts pid=37556 resp_time=532
5:41PM WRN Slow request method=POST path=/posts pid=37556 resp_time=532
5:41PM INF Access method=GET path=/users pid=37556 resp_time=10
5:41PM ERR Database connection lost error="connection reset by peer" database=myapp pid=37556
```

Source: <https://github.com/rs/zerolog/blob/4f50ae2ed060b877dac885c21dedad0e0c19e875/pretty.png>





# Structured logging

The general idea of structured logging is to format it as a collection of key-value pairs. Where the *key* should make crystal clear the meaning of the *value*.

It should be easy for computers to parse, transform and store it. The most common format is JSON, However there are others, such as strings following a well defined pattern.

There are standards for how to format or structure the log and for keys and their meaning:

- [Elastic Common Schema \(ECS\)](#)
- [Common Log Format](#)
- [Extended Log Format](#)
- [Windows Event Log](#)

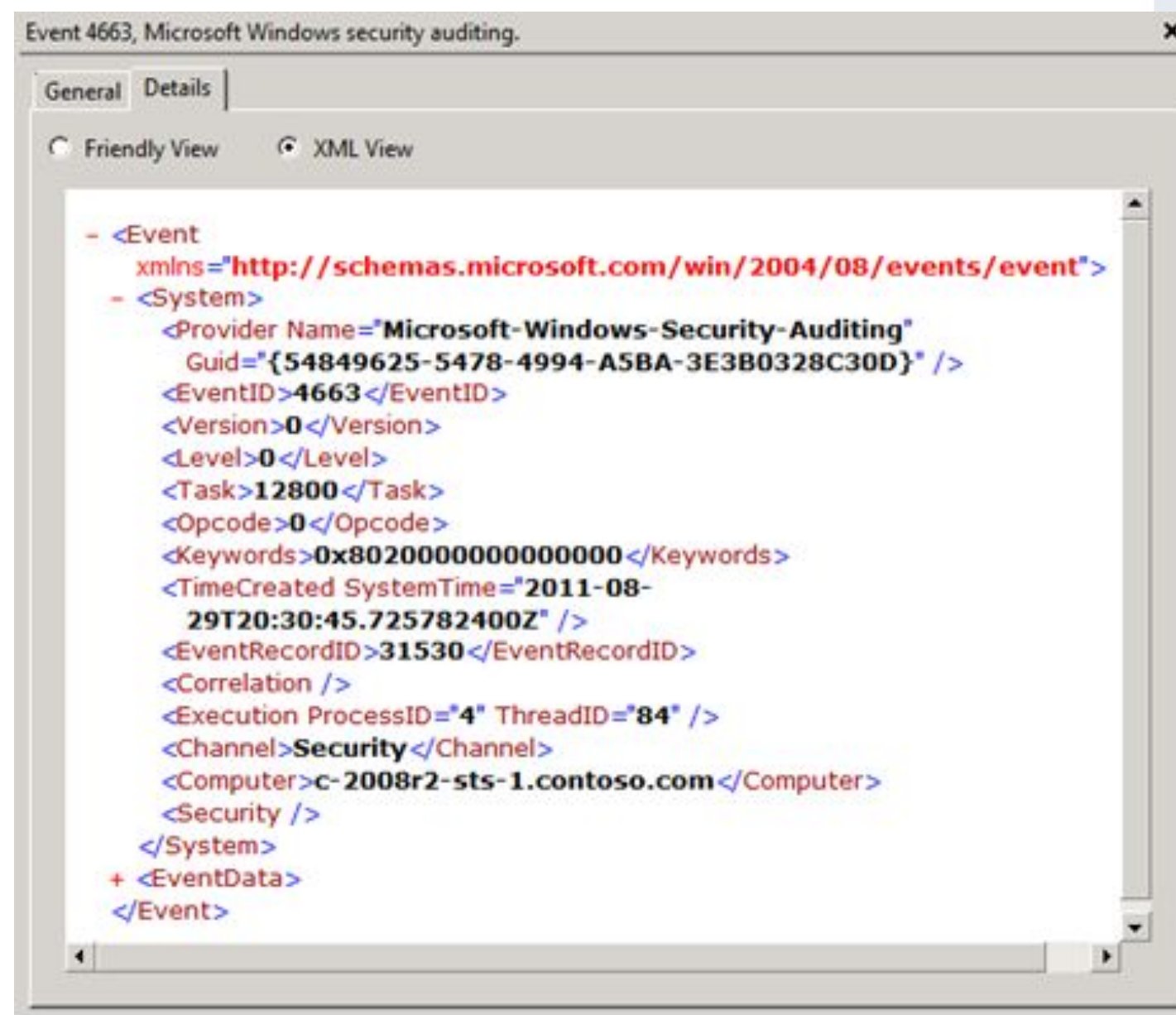
**LogFormat** "%h %l %u %t \"%r\" %>s %b"

127.0.0.1 user-identifier frank [10/Oct/2000:13:55:36 -0700] "GET /apache\_pb.gif HTTP/1.0" 200 2326

A dash ( - ) in a field indicates missing data.

- 127.0.0.1 is the IP address of the client (remote host) which made the request to the server.
- user-identifier is the RFC 1413 [identity](#) of the client. Usually "-".
- frank is the userid of the person requesting the document. Usually "-" unless [.htaccess](#) has requested authentication.
- [10/Oct/2000:13:55:36 -0700] is the date, time, and time zone that the request was received, by default in [strftime](#) format %d/%b/%Y:%H:%M:%S %z .
- "GET /apache\_pb.gif HTTP/1.0" is the request line from the client. The method GET , /apache\_pb.gif the resource requested, and HTTP/1.0 the [HTTP protocol](#).
- 200 is the [HTTP status code](#) returned to the client. 2xx is a successful response, 3xx a redirection, 4xx a client error, and 5xx a server error.
- 2326 is the size of the object returned to the client, measured in [bytes](#).

```
{
  "log.level": "info",
  "@timestamp": "2023-07-05T20:09:05.419Z",
  "message": "Non-zero metrics in the last 30s",
  "component": {
    "binary": "metricbeat",
    "dataset": "elastic_agent.metricbeat",
    "id": "http/metrics-monitoring",
    "type": "http/metrics"
  },
  "log": {
    "source": "http/metrics-monitoring"
  },
  "log.logger": "monitoring",
  "log.origin": {
    "file.line": 187,
    "file.name": "log/log.go"
  },
  "service.name": "metricbeat",
  "ecs.version": "1.6.0"
}
```





# Elastic Common Schema (ECS)

ECS field	Description	Example
@timestamp	The timestamp of the log event.	"2019-08-06T12:09:12.375Z"
log.level	The level or severity of the log event.	"INFO"
log.logger	The name of the logger inside an application.	"org.example.MyClass"
log.origin.file.name	The name of the file containing the source code which originated the log event.	"App.java"
log.origin.file.line	The line number of the file containing the source code which originated the log event.	42
log.origin.function	The name of the function or method which originated the log event.	"methodName"
message	The log message.	"Hello World!"
error.type	Only present for logs that contain an exception or error. The type or class of the error if this log event contains an exception.	"java.lang.NullPointerException"
error.message	Only present for logs that contain an exception or error. The message of the exception or error.	"The argument cannot be null"
error.stack_trace	Only present for logs that contain an exception or error. The full stack trace of the exception or error as a raw string.	"Exception in thread \"main\" java.lang.NullPointerException\n\tat org.example.App.methodName(App.java:42) "
process.thread.name	The name of the thread the event has been logged from.	"main"



# Elastic Common Schema (ECS)

[Platform](#)[Use cases](#)[Pricing](#)[Customers](#)[Resources](#)[Company](#)[Contact](#)[Login](#)

## Elastic Common Schema (ECS)

### Reference:

8.8 (current)

[Overview](#)[Using ECS](#)

### ECS Field Reference

[Base Fields](#)[Agent Fields](#)[Autonomous System Fields](#)[Client Fields](#)[Cloud Fields](#)[Code Signature Fields](#)[Container Fields](#)[Data Stream Fields](#)[Destination Fields](#)[Device Fields](#)[DLL Fields](#)[DNS Fields](#)[ECS Fields](#)[ELF Header Fields](#)[Email Fields](#)[Error Fields](#)[Event Fields](#)[FaaS Fields](#)[File Fields](#)[Geo Fields](#)[Group Fields](#)[Hash Fields](#)[Host Fields](#)[HTTP Fields](#)[Interface Fields](#)[Log Fields](#)[Mach-O Header Fields](#)[Elastic Docs](#) › [Elastic Common Schema \(ECS\) Reference \[8.8\]](#)

## ECS Field Reference

This is the documentation of ECS version 8.8.0.

ECS defines multiple groups of related fields. They are called "field sets". The [Base](#) field set is the only one whose fields are defined at the root of the event.

All other field sets are defined as objects in Elasticsearch, under which all fields are defined.

For a single page representation of all fields, please see the [generated CSV of fields](#).

### Field Sets

Field Set	Description
<a href="#">Base</a>	All fields defined directly at the root of the events.
<a href="#">Agent</a>	Fields about the monitoring agent.
<a href="#">Autonomous System</a>	Fields describing an Autonomous System (Internet routing prefix).
<a href="#">Client</a>	Fields about the client side of a network connection, used with server.
<a href="#">Cloud</a>	Fields about the cloud resource.
<a href="#">Code Signature</a>	These fields contain information about binary code signatures.
<a href="#">Container</a>	Fields describing the container that generated this event.
<a href="#">Data Stream</a>	The data_stream fields take part in defining the new data stream naming scheme.
<a href="#">Destination</a>	Fields about the destination side of a network connection, used with source.
<a href="#">Device</a>	Fields characterizing a (mobile) device a process or application is running on.

Source: <https://www.elastic.co/guide/en/ecs/8.8/ecs-field-reference.html>



# Elastic Common Schema (ECS)

## Tracing Fields



Distributed tracing makes it possible to analyze performance throughout a microservice architecture all in one view. This is accomplished by tracing all of the requests - from the initial web request in the front-end service - to queries made through multiple back-end services.

Unlike most field sets in ECS, the tracing fields are **not** nested under the field set name. In other words, the correct field name is `trace.id`, not `tracing.trace.id`, and so on.

### Tracing Field Details



Field	Description	Level
<code>span.id</code>	<p>Unique identifier of the span within the scope of its trace.</p> <p>A span represents an operation within a transaction, such as a request to another service, or a database query.</p> <p>type: keyword</p> <p>example: <code>3ff9a8981b7ccd5a</code></p>	extended
<code>trace.id</code>	<p>Unique identifier of the trace.</p> <p>A trace groups multiple events like transactions that belong together. For example, a user request handled by multiple inter-connected services.</p> <p>type: keyword</p> <p>example: <code>4bf92f3577b34da6a3ce929d0e0e4736</code></p>	extended
<code>transaction.id</code>	<p>Unique identifier of the transaction within the scope of its trace.</p> <p>A transaction is the highest level of work measured within a service, such as a request to a server.</p> <p>type: keyword</p> <p>example: <code>00f067aa0ba902b7</code></p>	extended

# Elastic Common Schema (ECS)

**Elastic contributes Elastic Common Schema (ECS) to OpenTelemetry (OTel), helping accelerate adoption of OTel-based observability and security**

By Ken Exner

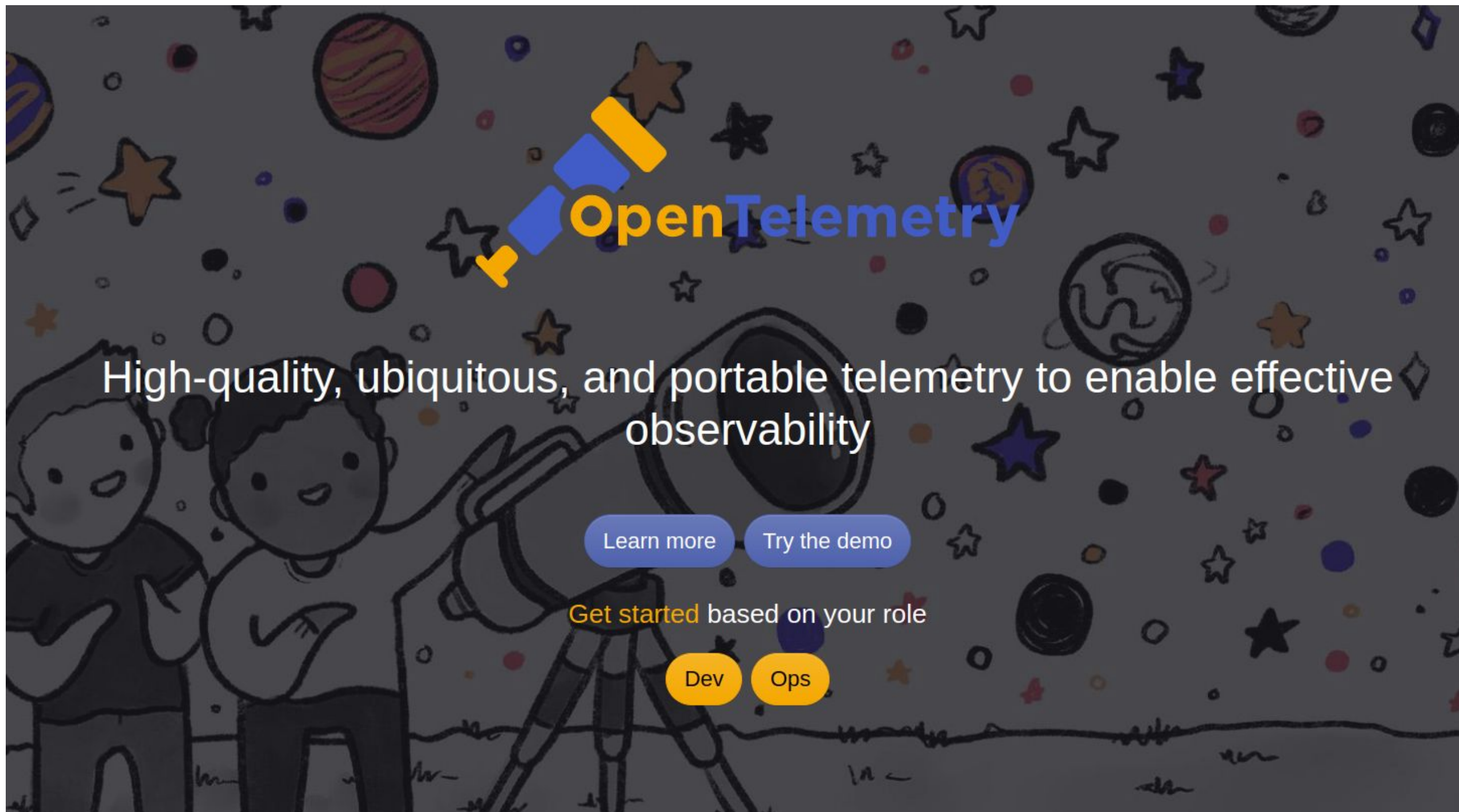
18 April 2023



- [ECS Getting started](#)
- [ECS and OpenTelemetry](#)

Source: <https://www.elastic.co/guide/en/ecs/8.8/ecs-field-reference.html>





OpenTelemetry is a collection of APIs, SDKs, and tools. Use it to instrument, generate, collect, and export telemetry data (metrics, logs, and traces) to help you analyze your software's performance and behavior.

OpenTelemetry is **generally available** across [several languages](#) and is suitable for use.

<https://opentelemetry.io>

# Ingesting and storing logs

# Ingest and store



## Collect the logs

Specialised programs can harvest log files or directly read the output of an application. The application can send them directly to storage or to a collector



## Enhance and transform

Add metadata, environment information, parse unstructured/non-JSON entries to JSON, standardise key naming.



## Store

Send the logs to a storage, usually some noSQL-like storage made for search, like Elasticsearch






# Ingest and store

## Some ways to ingest logs

- Elastic Agent
- Filebeat
- Logstash
- Fluentd
- Fluent Bit
- Splunk Forwarder
- OpenTelemetry client libs and collector

**Search and visualize**



# Search and visualize

Once the logs are stored, and ideally all in the same format, they can be searched. The most common is the storage be also the search engine.

They work together with other tools providing a user friendly interface to query and visualize the results.

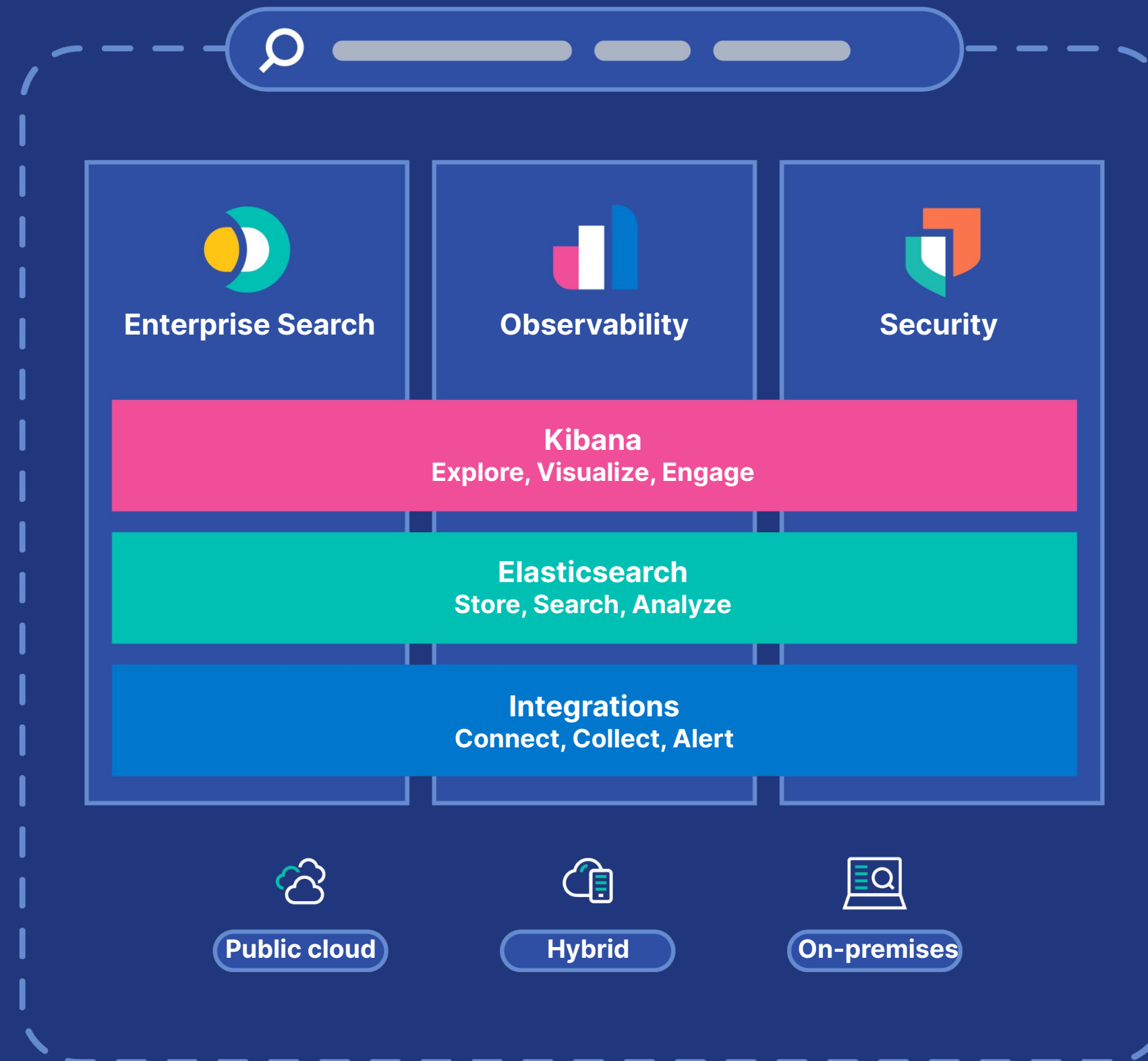
Some options to query and visualize logs:

- Kibana
- Graylog
- Grafana



# Demo\* (with quite some bias) The Elastic Search Platform

\* the environment where the demo was has been terminated. Thus I removes the link.



**Thank you  
Questions?**

# Contact



Anderson Queiroz | Sr. Software  
Engineer

[me@andersonq.me](mailto:me@andersonq.me) | [anderson.queiroz@elastic.co](mailto:anderson.queiroz@elastic.co)

<https://www.linkedin.com/in/andersonq/>

<https://github.com/AndersonQ/talks>