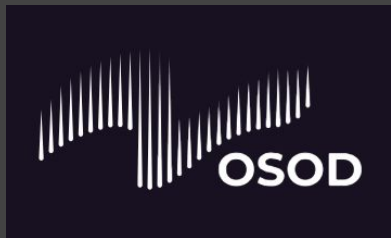


Get a Grip on your Telemetry using the OpenTelemetry Collector

Adam Gardner | APAC Lead Developer Advocate at Dynatrace



Agenda

1. What Problem are we trying to solve?
2. Overview of OpenTelemetry and the Collector
3. Scenarios
 - a. Basic Setup w/ Batching
 - b. Add File and OS Info
 - c. Setting default log severity
 - d. Log severity based on content
 - e. Content based K/V enrichment
 - f. Log line standardisation
 - g. Sensitive info removal (dropping / filtering logs)
 - h. Dropping low value logs
 - i. Log enrichment: Ownership information
 - j. Extracting metrics from logs
 - k. Enriching logs from CSV files



Key Takeaway

The OpenTelemetry Collector is a MUST HAVE component in a modern Observability stack.



What's the Problem?



OpenTelemetry (it's a lot of stuff!)

Instrumentation (APIs, SDK, Auto instrumentation)

OpenTelemetry Collector

OpenTelemetry Operator

OpAMP

Semantic Conventions



Hands on

OpenTelemetry (it's a lot of stuff!)

Instrumentation (APIs, SDK, Auto instrumentation)

OpenTelemetry Collector

OpenTelemetry Operator

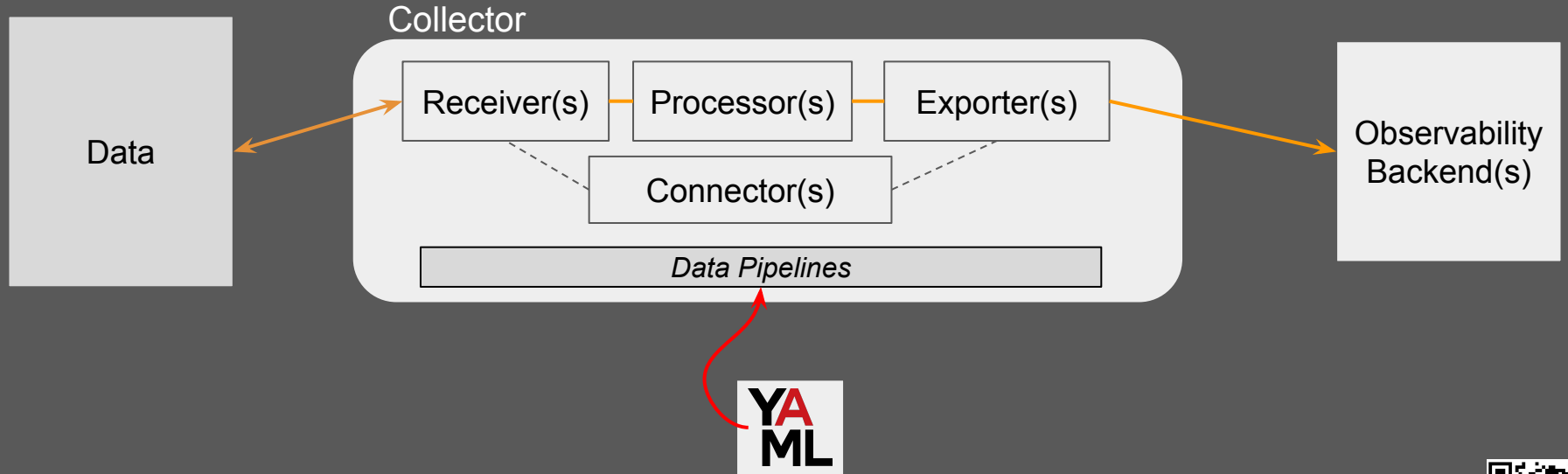
OpAMP

Semantic Conventions



Hands on

Collector Architecture



Receivers

- Hundreds of receivers (you can write your own too) <https://dt-url.net/otel-receivers>
 - OS stats
 - Log Files 
 - Databases
 - Cloud Vendors
 - Kubernetes
 - Queues
 - Webhooks
 - Syslog / Netflow
 - Prometheus



Processors

- Many ways to process data (you can write your own too) <https://dt-url.net/otel-processors>
 - Filtering / Redaction 
 - Resource Detection 
 - GeoIP Lookups
 - CSV Lookups 
 - Data Transformation 



Exporters

- Most vendors have standardised on the OpenTelemetry Protocol (OTLP)

<https://dt-url.net/otel-exporters>



Basic Setup

receivers:

filelog:

include: file.log



processors:

batch:

send_batch_size: 500



timeout: 2s

exporters:

debug:

verbosity: detailed



otlphttp/dynatrace:

endpoint: "[https://abc12345.live.dynatrace.com/api/v2/otlp](\"https://abc12345.live.dynatrace.com/api/v2/otlp\")"

headers:

Authorization: "Api-Token \${env:DT_API_TOKEN}"

service:

pipelines:

logs:

receivers: [filelog]

processors: [batch]

exporters: [debug, otlphttp/dynatrace]



```
file.log  U ●  
  
file.log  
1  My first log line
```

1 record

Executed at: 10/20/2025, 11:02:45 ⓘ

timestamp

content

10/20/2025, 11:02:46 AM

My first log line



Hands on

Add File and OS Info

```
receivers:  
  filelog:  
    include: file.log  
    include_file_name: true  
    Include_file_path: true
```

```
processors:  
  batch:  
    send_batch_size: 500  
    timeout: 2s  
  resourcedetection:  
    detectors: ["system"]  
    system:  
      hostname_sources: ["os"]
```

```
exporters:  
  debug:  
    verbosity: detailed
```

```
otlhttp/dynatrace:  
  endpoint: "https://abc12345.live.dynatrace.com/api/v2/otlp"  
  headers:  
    Authorization: "Api-Token ${env:DT_API_TOKEN}"
```

```
service:  
  pipelines:  
    logs:  
      receivers: [filelog]  
      processors: [resourcedetection, batch]  
      exporters: [debug, otlhttp/dynatrace]
```

timestamp	content	host.name	loglevel	status	log.file.name	log.file.path	os.type
10/20/2025, 11:09:14 AM	My second log line	adams-macbook-air	NONE	NONE	file.log	/home/username/file.log	linux

```
file.log  U ●  
  
file.log  
1  
2   My first log line  
3   My second log line
```



Log Severity

receivers:

filelog:

include: file.log
include_file_name: true
include_file_path: true

processors:

batch:

send_batch_size: 500
timeout: 2s

resourcedetection:

detectors: ["system"]

system:

hostname_sources: ["os"]

transform:

log_statements:

- context: log
statements:
- set(log.severity_text, "INFO") where !sMatch(log.severity_text, "")
- set(log.severity_number, SEVERITY_NUMBER_INFO) where(log.severity_text, "INFO")

exporters:

debug:

verbosity: detailed

otlhttp/dynatrace:

endpoint: "https://abc12345.live.dynatrace.com/api/v2/otlp"

headers:

Authorization: "Api-Token \${env:DT_API_TOKEN}"

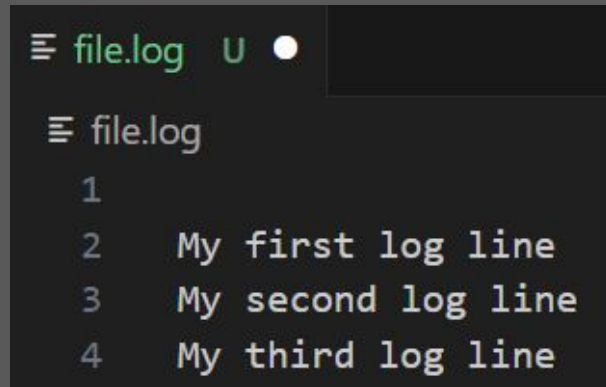
service:

pipelines:

logs:

receivers: [filelog]
processors: [resourcedetection, transform, batch]
exporters: [debug, otlhttp/dynatrace]

timestamp	content	host.name	loglevel	status	log.file.name	log.file.path	os.type
10/20/2025, 11:09:14 AM	My second log line	adams-macbook-air	NONE	NONE	file.log	/home/username/file.log	linux



timestamp	content	host.name	loglevel	status	log.file.name	log.file.path	os.type
10/20/2025, 11:15:14 AM	My third log line	adams-macbook-air	INFO	INFO	file.log	/home/username/file.log	linux



Hands on



Hands on

Content based Log Severity

```
file.log
1
2   My first log line
3   My second log line
4   My third log line
5   My fourth dummy log line. Please investigate - something is broken.
6   My fifth dummy log line
```

processors:

transform:

log_statements:

- context: log

statements:

- set(log.severity_text, "INFO") where IsMatch(log.severity_text, "")
- set(log.severity_number, SEVERITY_NUMBER_INFO) where (log.severity_text, "INFO")
- set(log.severity_number, SEVERITY_NUMBER_INFO) where IsMatch(log.severity_text, "INFO")
- set(log.severity_text, "ERROR") where IsMatch(body, ".*something is broken.*")
- set(log.severity_number, SEVERITY_NUMBER_ERROR) where IsMatch(body, ".*something is broken.*")

2 records Executed at: 10/20/2025, 11:22:15 ⓘ

timestamp	content	host.name	loglevel	status	log.file.name	log.file.path	os.type
10/20/2025, 11:22:15 AM	My fourth dummy log line. Please investigate - something is broken.	adams-macbook-air	ERROR	ERROR	file.log	/home/username/file.log	linux
10/20/2025, 11:22:15 AM	My fifth dummy log line	adams-macbook-air	INFO	INFO	file.log	/home/username/file.log	linux

Content based Conditional Metadata

≡ file.log

1 My dummy log line from userId=123 part of userTier=tier1

1 record Executed at: 10/20/2025, 15:14:39 ⓘ

timestamp	content	support.tier	loglevel	status
10/20/2025, 3:14:40 PM	My dummy log line from userId=123 part of userTier=tier1	gold	INFO	INFO

processors:

batch:

send_batch_size: 500

timeout: 2s

resourcedetection:

detectors: ["system"]

system:

hostname_sources: ["os"]

transform:

log_statements:

- context: log

statements:

- set(log.attributes["support.tier"], "gold") where IsMatch(body, ".*userTier=tier1.*")
- set(log.attributes["support.tier"], "silver") where IsMatch(body, ".*userTier=tier2.*")
- set(log.attributes["support.tier"], "bronze") where IsMatch(body, ".*userTier=tier3.*")



Hands on



Hands on

Standardising Log Lines w/ Zero Code Changes

timestamp	content	host.name	loglevel	status	log.file.name	log.file.path	os.type
10/20/2025, 11:36:34 AM	My dummy log line from userId=566 part of user.tier=tier3	adams-macbook-air	INFO	INFO	file.log	/home/username/file.log	linux

```
processors:  
  batch:  
    send_batch_size: 500  
    timeout: 2s  
  resourcedetection:  
    detectors: ["system"]  
  system:  
    hostname_sources: ["os"]  
transform:  
  log_statements:  
    - context: log  
      statements:  
        - replace_pattern(body, "user.tier=", "userTier=")  
        - set(log.attributes["support.tier"], "gold") where IsMatch(body, ".*userTier=tier1.*")  
        - set(log.attributes["support.tier"], "silver") where IsMatch(body, ".*userTier=tier2.*")  
        - set(log.attributes["support.tier"], "bronze") where IsMatch(body, ".*userTier=tier3.*")
```

timestamp	content	support.tier	host.name	loglevel	status	log.file.name	log.file.path	os.type
10/20/2025, 11:40:44 AM	My dummy log line from userId=566 part of userTier=tier3	bronze	adams-macbook-air	INFO	INFO	file.log	/home/username/file.log	linux

Processing Sensitive Log Lines

```
≡ file.log  
1 A dummy log line. The password is abc123
```

Redaction

transform:

log_statements:

- context: log

statements:

- replace_pattern(log.body, "The password is .*", "*****")

1 record Executed at: 10/20/2025, 15:21:19 ⓘ

timestamp	content	loglevel	status
10/20/2025, 3:21:20 PM	INFO A dummy log line. *****	INFO	INFO



Hands on



Hands on

Processing Sensitive Log Lines

```
≡ file.log  
1 A dummy log line. The password is abc123
```

Removal (Filtering [Out])

processors:

filter:

error_mode: ignore

logs:

log_record:

- 'IsMatch(body, ".*(?i)password.*") # Case insensitive match for any log line containing 'password'

service:

pipelines:

logs:

receivers: [filelog]

processors: [filter, resourcedetection, transform, batch]

Ordering matters.
Filter to remove first.

Removal of Low Value Logs

SeverityNumber range	Range name	Meaning
1-4	TRACE	A fine-grained debugging event. Typically disabled in default configurations.
5-8	DEBUG	A debugging event.
9-12	INFO	An informational event. Indicates that an event happened.
13-16	WARN	A warning event. Not an error but is likely more important than an informational event.
17-20	ERROR	An error event. Something went wrong.
21-24	FATAL	A fatal error such as application or system crash.

processors:

filter:

error_mode: ignore

logs:

log_record:

- 'severity_number < SEVERITY_NUMBER_WARN'



Hands on



Organisational Enrichment

We know file.log is owned by a particular team.
How do we route tickets to them?
How do we cross charge them for log storage?

Requirement:

- Every log line from file.log must include:
 - Team name
 - Team Contact Details
 - Team cross charge code

```
receivers:  
filelog:  
  include: file.log  
  include_file_name: true  
  include_file_path: true  
attributes:  
  team.name: teamA  
  team.email: teamA@example.com  
  team.chargecode: ABC556D
```

2 records Executed at: 10/20/2025, 13:08:43 ⓘ

timestamp	content	team.name	team.email	team.chargecode	host.name	loglevel	status	log.file.name	log.file.path	os.type
10/20/2025, 1:08:44 PM	My first dummy log line	teamA	teamA@example.com	ABC556D	adams-macbook-air	INFO	INFO	file.log	/home/username/file.log	linux
10/20/2025, 1:08:54 PM	My second dummy log line	teamA	teamA@example.com	ABC556D	adams-macbook-air	INFO	INFO	file.log	/home/username/file.log	linux

Transforming Logs to Metrics

```
≡ file.log
```

```
1 A first log line field=200
```

1. Read log lines
2. Extract the value of “field”
3. Sum it up & send as a metric

```
service:  
  pipelines:  
    logs:  
      receivers: [filelog]  
      exporters: [signalmetrics] ← 3  
  metrics:  
    receivers: [signalmetrics] ← 4  
    exporters: [otlphhttp, debug]
```

```
receivers:  
  filelog:  
    include: file.log  
    operators:  
      - type: regex_parser  
        regex: 'field=(?<field_value>\d+)' ← 1  
        parse_from: body  
connectors:  
  signalmetrics:  
    logs:  
      - name: "field_value"  
        description: "The value of field from logs"  
        sum:  
          value: Int(attributes["field_value"]) ← 2
```



Enriching Logs using CSV Files

≡ file.log

```
1 INFO hello world component=frontend
2 INFO hello world component=middleware
3 INFO hello world 2 component=backend
```

example.csv

```
1 component,responsible_team,owner,support_tier
2 frontend,teamA,alice@example.com,tier1
3 backend,teamB,bob@example.com,tier2
```

3 records

Executed at: 10/20/2025, 14:37:23 ⓘ

timestamp	content	component	responsible_team	owner	support_tier	loglevel	status
10/20/2025, 2:37:24 PM	INFO hello world component=frontend	frontend	teamA	alice@example.com	tier1	INFO	INFO
10/20/2025, 2:37:24 PM	INFO hello world component=middleware	middleware	null	null	null	INFO	INFO
10/20/2025, 2:37:24 PM	INFO hello world 2 component=backend	backend	teamB	bob@example.com	tier2	INFO	INFO



Enriching Logs using CSV Files

receivers:

filelog:

include: file.log

operators:

- type: regex_parser

regex: 'component=(?<component>\d+)'

parse_from: body

processors:

lookup:

csv: "example.csv"

context: attributes

field: component

service:

pipelines:

logs:

receivers: [filelog]

processors: [lookup]

exporters: [debug, otlphttp]

3 records Executed at: 10/20/2025, 14:37:23 ⓘ

timestamp	content	component	responsible_team	owner	support_tier	loglevel	status
10/20/2025, 2:37:24 PM	INFO hello world component=frontend	frontend	teamA	alice@example.com	tier1	INFO	INFO
10/20/2025, 2:37:24 PM	INFO hello world component=middleware	middleware	null	null	null	INFO	INFO
10/20/2025, 2:37:24 PM	INFO hello world 2 component=backend	backend	teamB	bob@example.com	tier2	INFO	INFO



Summary

- The collector is an enormously powerful component
- It should be part of every 2025 Observability setup
- Try all of this in a browser: <https://dt-url.net/demo-collector-patterns>
- Lookup processor: <https://www.youtube.com/watch?v=zM43M5dzKpk>

Find me: <https://linkedin.com/in/agardner1>



Thank You!



Adam Gardner | APAC Lead Developer Advocate at Dynatrace