```
# 🔧 OpenPipeline Processing

> **Series:** OPLOGS | **Notebook:** 3 of 8 | **Created:** December 2025

## Configuring Pipeline Stages for Log Transformation

This notebook covers OpenPipeline processing stages: parsing, enrichment,
metric extraction, event generation, bucket routing, and filtering.

---

## Table of Contents

1. Parsing & Field Extraction
2. Metric Extraction from Logs
3. Attribute Creation & Enrichment
4. Event Generation from Logs
5. Bucket Routing
6. Filtering & Sampling
7. Complete Pipeline Example


## Prerequisites

- ✅ Access to a Dynatrace environment with log data
- ✅ OpenPipeline configuration permissions
- ✅ Completed OPLOGS-01 and OPLOGS-02



## 2. Parsing & Field Extraction

Parsing extracts structured fields from unstructured log content **at
ingestion time**.

### DPL (Dynatrace Pattern Language) Matchers

![DPL Matchers]
(
ciIHZpZXdCb3g9IjAgMCA4MDAgMzgwIj4KICA8ZGVmcz4KICAgIDxsaW5lYXJHcmFkaWVudCBpZD0
idGV4dEdyYWQiIHgxPSIwJSIgeTE9IjAlIiB4Mj0iMTAwJSIgeTI9IjEwMCUiPgogICAgICA8c3Rv
cCBvZmZzZXQ9IjAlIiBzdHlsZT0ic3RvcC1jb2xvcjojM2I4MmY2O3N0b3Atb3BhY2l0eToxIiAvP
gogICAgICA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiMyNTYzZWI7c3RvcC
1vcGFjaXR5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50PgogICAgPGxpbmVhckdyYWRpZW50IGl
kPSJudW1HcmFkIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMDAlIj4KICAgICAgPHN0b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3IzEwYjk4MTtzdG9wLW9wYWNpdHk6MSIgL
z4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcjojMDU5NjY5O3N0b3
```

Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxsaW5lYXJHcmFkaWVudCB
pZD0ibmV0R3JhZCIgeDE9IjAlIiB5MT0iMCUiIHgyPSIxMDAlIiB5Mj0iMTAwJSI+CiAgICAgIDxz
dG9wIG9mZnNldD0iMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiNmNTllMGI7c3RvcC1vcGFjaXR5OjEiI
C8+CiAgICAgIDxzdG9wIG9mZnNldD0iMTAwJSIgc3R5bGU9InN0b3AtY29sb3I6I2Q5NzcwNjtzdG
9wLW9wYWNpdHk6MSIgLz4KICAgIDwvbGluZWFyR3JhZGllbnQ+CiAgICA8bGluZWFyR3JhZGllbnQ
gaWQ9InRpbWVHcmFkIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIxMDAlIj4KICAgICAg
PHN0b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6IzhiNWNmNjtzdG9wLW9wYWNpdHk6M
SIgLz4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcjojN2MzYWVkO3
N0b3Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxmaWx0ZXIgaWQ9ImR
wbFNoYWRvdyI+CiAgICAgIDxmZURyb3BTaGFkb3cgZHg9IjEiIGR5PSIxIiBzdGREZXZpYXRpb249
IjIiIGZsb29kLW9wYWNpdHk9IjAuMTUiLz4KICAgIDwvZmlsdGVyPgogIDwvZGVmcz4KCiAgPCEtL
SBCYWNrZ3JvdW5kIC0tPgogIDxyZWN0IHdpZHRoPSI4MDAiIGhlaWdodD0iMzgwIiBmaWxsPSIjZj
hmOWZhIiByeD0iMTAiLz4KCiAgPCEtLSBUaXRsZSAtLT4KICA8dGV4dCB4PSI0MDAiIHk9IjI4IiB
mb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTgiIGZvbnQtd2VpZ2h0
PSJib2xkIiBmaWxsPSIjMzMzIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5EUEwgUGF0dGVybiBNYXRja
GVycyBRdWljayBSZWZlcmVuY2U8L3RleHQ+CiAgPHRleHQgeD0iNDAwIiB5PSI0OCIgZm9udC1mYW
1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjExIiBmaWxsPSIjNjY2IiB0ZXh0LWF
uY2hvcj0ibWlkZGxlIj5EeW5hdHJhY2UgUGF0dGVybiBMYW5ndWFnZSBmb3IgTG9nIFBhcnNpbmc8
L3RleHQ+CgogIDwhLS0gVGV4dCBNYXRjaGVycyBTZWN0aW9uIC0tPgogIDxyZWN0IHg9IjMwIiB5P
SI2NSIgd2lkdGg9IjM2NSIgaGVpZ2h0PSIxNDAiIHJ4PSIxMCIgZmlsbD0iI2ZmZiIgc3Ryb2tlPS
IjZTJlOGYwIiBzdHJva2Utd2lkdGg9IjIiLz4KICA8cmVjdCB4PSIzMCIgeT0iNjUiIHdpZHRoPSI
zNjUiIGhlaWdodD0iMzAiIHJ4PSIxMCIgZmlsbD0idXJsKCN0ZXh0R3JhZCkiLz4KICA8dGV4dCB4
PSIyMTIiIHk9Ijg1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iM
TIiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+VG
V4dCBNYXRjaGVyczwvdGV4dD4KCiAgPHRleHQgeD0iNTAiIHk9IjExNSIgZm9udC1mYW1pbHk9Im1
vbm9zcGFjZSIgZm9udC1zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9IiMyNTYzZWIi
PkxEPC90ZXh0PgogIDx0ZXh0IHg9IjEwMCIgeT0iMTE1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhb
nMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiMzMzMiPkxpbmUgZGF0YSAoZ3JlZWR5KTwvdG
V4dD4KICA8dGV4dCB4PSI1MCIgeT0iMTM1IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXN
pemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0iIzI1NjNlYiI+V09SRDwvdGV4dD4KICA8
dGV4dCB4PSIxMDAiIHk9IjEzNSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250L
XNpemU9IjEwIiBmaWxsPSIjMzMzIj5TaW5nbGUgd29yZDwvdGV4dD4KICA8dGV4dCB4PSI1MCIgeT
0iMTU1IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD0
iYm9sZCIgZmlsbD0iIzI1NjNlYiI+REFUQTwvdGV4dD4KICA8dGV4dCB4PSIxMDAiIHk9IjE1NSIg
Zm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjMzMzI
j5BbnkgdGV4dCAobm9uLWdyZWVkeSk8L3RleHQ+CiAgPHRleHQgeD0iNTAiIHk9IjE3NSIgZm9udC
1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw
9IiMyNTYzZWIiPk5TUEFDRTwvdGV4dD4KICA8dGV4dCB4PSIxMDAiIHk9IjE3NSIgZm9udC1mYW1p
bHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjMzMzIj5Ob24td2hpd
GVzcGFjZTwvdGV4dD4KICA8dGV4dCB4PSI1MCIgeT0iMTk1IiBmb250LWZhbWlseT0ibW9ub3NwYW
NlIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0iIzI1NjNlYiI+U1BBQ0U
8L3RleHQ+CiAgPHRleHQgeD0iMTAwIiB5PSIxOTUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1z
ZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzMzMyI+V2hpdGVzcGFjZSBjaGFyYWN0ZXJzPC90Z
Xh0PgoKICA8dGV4dCB4PSIyMjAiIHk9IjExNSIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC
1zaXplPSIxMCIgZmlsbD0iIzY0NzQ4YiI+ImVycm9yIGluIG1vZHVsZSI8L3RleHQ+CiAgPHRleHQ
geD0iMjIwIiB5PSIxMzUiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZp
bGw9IiM2NDc0OGIiPiJFUlJPUiI8L3RleHQ+CiAgPHRleHQgeD0iMjIwIiB5PSIxNTUiIGZvbnQtZ
mFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NDc0OGIiPiJhbnkgY29udG

VudCI8L3RleHQ+CiAgPHRleHQgeD0iMjIwIiB5PSIxNzUiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2U
iIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NDc0OGIiPiJ1c2VyQGhvc3QuY29tIjwvdGV4dD4KICA8
dGV4dCB4PSIyMjAiIHk9IjE5NSIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxM
CIgZmlsbD0iIzY0NzQ4YiI+IiAgICIgKHRhYnMvc3BhY2VzKTwvdGV4dD4KCiAgPCEtLSBOdW1iZX
IgTWF0Y2hlcnMgU2VjdGlvbiAtLT4KICA8cmVjdCB4PSI0MDUiIHk9IjY1IiB3aWR0aD0iMzY1IiB
oZWlnaHQ9IjE0MCIgcng9IjEwIiBmaWxsPSIjZmZmIiBzdHJva2U9IiNlMmU4ZjAiIHN0cm9rZS13
aWR0aD0iMiIvPgogIDxyZWN0IHg9IjQwNSIgeT0iNjUiIHdpZHRoPSIzNjUiIGhlaWdodD0iMzAiI
HJ4PSIxMCIgZmlsbD0idXJsKCNudW1HcmFkKSIvPgogIDx0ZXh0IHg9IjU4NyIgeT0iODUiIGZvbn
QtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMiIgZm9udC13ZWlnaHQ9ImJ
vbGQiIGZpbGw9IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5OdW1iZXIgTWF0Y2hlcnM8L3Rl
eHQ+CgogIDx0ZXh0IHg9IjQyNSIgeT0iMTE1IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250L
XNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0iIzA1OTY2OSI+SU5UPC90ZXh0PgogID
x0ZXh0IHg9IjQ3NSIgeT0iMTE1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQ
tc2l6ZT0iMTAiIGZpbGw9IiMzMzMiPkludGVnZXI8L3RleHQ+CiAgPHRleHQgeD0iNDI1IiB5PSIx
MzUiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTEiIGZvbnQtd2VpZ2h0PSJib
2xkIiBmaWxsPSIjMDU5NjY5Ij5MT05HPC90ZXh0PgogIDx0ZXh0IHg9IjQ3NSIgeT0iMTM1IiBmb2
50LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiMzMzMiPkx
vbmcgaW50ZWdlcjwvdGV4dD4KICA8dGV4dCB4PSI0MjUiIHk9IjE1NSIgZm9udC1mYW1pbHk9Im1v
bm9zcGFjZSIgZm9udC1zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9IiMwNTk2NjkiP
kRPVUJMRTwvdGV4dD4KICA8dGV4dCB4PSI0NzUiIHk9IjE1NSIgZm9udC1mYW1pbHk9IkFyaWFsLC
BzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjMzMzIj5EZWNpbWFsIG51bWJlcjwvdGV
4dD4KICA8dGV4dCB4PSI0MjUiIHk9IjE3NSIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1z
aXplPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9IiMwNTk2NjkiPkJPT0xFQU48L3RleHQ+C
iAgPHRleHQgeD0iNDk1IiB5PSIxNzUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm
9udC1zaXplPSIxMCIgZmlsbD0iIzMzMyI+dHJ1ZS9mYWxzZTwvdGV4dD4KICA8dGV4dCB4PSI0MjU
iIHk9IjE5NSIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMSIgZm9udC13ZWln
aHQ9ImJvbGQiIGZpbGw9IiMwNTk2NjkiPkhFWDwvdGV4dD4KICA8dGV4dCB4PSI0NzUiIHk9IjE5N
SIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjMz
MzIj5IZXhhZGVjaW1hbDwvdGV4dD4KCiAgPHRleHQgeD0iNTk1IiB5PSIxMTUiIGZvbnQtZmFtaWx
5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NDc0OGIiPi0xLCA0MiwgMTAwMDwv
dGV4dD4KICA8dGV4dCB4PSI1OTUiIHk9IjEzNSIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9ud
C1zaXplPSIxMCIgZmlsbD0iIzY0NzQ4YiI+OTIyMzM3MjAzNjg1NDc3NTgwNzwvdGV4dD4KICA8dG
V4dCB4PSI1OTUiIHk9IjE1NSIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMCI
gZmlsbD0iIzY0NzQ4YiI+My4xNCwgLTAuNTwvdGV4dD4KICA8dGV4dCB4PSI1OTUiIHk9IjE3NSIg
Zm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzY0NzQ4YiI+dHJ1Z
SwgZmFsc2U8L3RleHQ+CiAgPHRleHQgeD0iNTk1IiB5PSIxOTUiIGZvbnQtZmFtaWx5PSJtb25vc3
BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NDc0OGIiPjB4MUYsIERFQURCRUVGPC90ZXh0Pgo
KICA8IS0tIE5ldHdvcmsvSWRlbnRpdHkgTWF0Y2hlcnMgU2VjdGlvbiAtLT4KICA8cmVjdCB4PSIz
MCIgeT0iMjE1IiB3aWR0aD0iMzY1IiBoZWlnaHQ9IjE0MCIgcng9IjEwIiBmaWxsPSIjZmZmIiBzd
HJva2U9IiNlMmU4ZjAiIHN0cm9rZS13aWR0aD0iMiIvPgogIDxyZWN0IHg9IjMwIiB5PSIyMTUiIH
dpZHRoPSIzNjUiIGhlaWdodD0iMzAiIHJ4PSIxMCIgZmlsbD0idXJsKCNuZXRHcmFkKSIvPgogIDx
0ZXh0IHg9IjIxMiIgeT0iMjM1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQt
c2l6ZT0iMTIiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZ
GRsZSI+TmV0d29yayAvIElkZW50aXR5PC90ZXh0PgoKICA8dGV4dCB4PSI1MCIgeT0iMjY1IiBmb2
50LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZml
sbD0iI2Q5NzcwNiI+SVBBRERSPC90ZXh0PgogIDx0ZXh0IHg9IjExMCIgeT0iMjY1IiBmb250LWZh
bWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiMzMzMiPklQdjQvS
VB2NiBhZGRyZXNzPC90ZXh0PgogIDx0ZXh0IHg9IjUwIiB5PSIyODUiIGZvbnQtZmFtaWx5PSJtb2

5vc3BhY2UiIGZvbnQtc2l6ZT0iMTEiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSIjZDk3NzA2Ij5
JUFY0PC90ZXh0PgogIDx0ZXh0IHg9IjExMCIgeT0iMjg1IiBmb250LWZhbWlseT0iQXJpYWwsIHNh
bnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiMzMzMiPklQdjQgb25seTwvdGV4dD4KICA8d
GV4dCB4PSI1MCIgeT0iMzA1IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjExIi
Bmb250LXdlaWdodD0iYm9sZCIgZmlsbD0iI2Q5NzcwNiI+SVBWNjwvdGV4dD4KICA8dGV4dCB4PSI
xMTAiIHk9IjMwNSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEw
IiBmaWxsPSIjMzMzIj5JUHY2IG9ubHk8L3RleHQ+CiAgPHRleHQgeD0iNTAiIHk9IjMyNSIgZm9ud
C1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbG
w9IiNkOTc3MDYiPkhUVFBVUkk8L3RleHQ+CiAgPHRleHQgeD0iMTE1IiB5PSIzMjUiIGZvbnQtZmF
taWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzMzMyI+SFRUUCBV
UkwgcGF0aDwvdGV4dD4KICA8dGV4dCB4PSI1MCIgeT0iMzQ1IiBmb250LWZhbWlseT0ibW9ub3NwY
WNlIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0iI2Q5NzcwNiI+VVVJRD
wvdGV4dD4KICA8dGV4dCB4PSIxMTAiIHk9IjM0NSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXN
lcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjMzMzIj5VVUlEIGZvcm1hdDwvdGV4dD4KCiAgPHRl
eHQgeD0iMjIwIiB5PSIyNjUiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiI
GZpbGw9IiM2NDc0OGIiPjEwLjAuMC4xLCA6OjE8L3RleHQ+CiAgPHRleHQgeD0iMjIwIiB5PSIyOD
UiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NDc0OGIiPjE
5Mi4xNjguMS4xMDA8L3RleHQ+CiAgPHRleHQgeD0iMjIwIiB5PSIzMDUiIGZvbnQtZmFtaWx5PSJt
b25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NDc0OGIiPjIwMDE6ZGI4OjoxPC90ZXh0P
gogIDx0ZXh0IHg9IjIyMCIgeT0iMzI1IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpem
U9IjEwIiBmaWxsPSIjNjQ3NDhiIj4vYXBpL3YxL3VzZXJzPC90ZXh0PgogIDx0ZXh0IHg9IjIyMCI
geT0iMzQ1IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNjQ3
NDhiIj5hMWIyYzNkNC0uLi48L3RleHQ+CgogIDwhLS0gVGltZS9EYXRlIE1hdGNoZXJzIFNlY3Rpb
24gLS0+CiAgPHJlY3QgeD0iNDA1IiB5PSIyMTUiIHdpZHRoPSIzNjUiIGhlaWdodD0iMTQwIiByeD
0iMTAiIGZpbGw9IiNmZmYiIHN0cm9rZT0iI2UyZThmMCIgc3Ryb2tlLXdpZHRoPSIyIi8+CiAgPHJ
lY3QgeD0iNDA1IiB5PSIyMTUiIHdpZHRoPSIzNjUiIGhlaWdodD0iMzAiIHJ4PSIxMCIgZmlsbD0i
dXJsKCN0aW1lR3JhZCkiLz4KICA8dGV4dCB4PSI1ODciIHk9IjIzNSIgZm9udC1mYW1pbHk9IkFya
WFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEyIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0id2
hpdGUiIHRleHQtYW5jaG9yPSJtaWRkbGUiPlRpbWUgLyBEYXRlPC90ZXh0PgoKICA8dGV4dCB4PSI
0MjUiIHk9IjI2NSIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMSIgZm9udC13
ZWlnaHQ9ImJvbGQiIGZpbGw9IiM3YzNhZWQiPlRJTUVTVEFNUDwvdGV4dD4KICA8dGV4dCB4PSI1M
TAiIHk9IjI2NSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIi
BmaWxsPSIjMzMzIj5JU08gdGltZXN0YW1wPC90ZXh0PgogIDx0ZXh0IHg9IjQyNSIgeT0iMjg1IiB
mb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIg
ZmlsbD0iIzdjM2FlZCI+REFURTwvdGV4dD4KICA8dGV4dCB4PSI0ODAiIHk9IjI4NSIgZm9udC1mY
W1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjMzMzIj5EYXRlIC
hZWVlZLU1NLUREKTwvdGV4dD4KICA8dGV4dCB4PSI0MjUiIHk9IjMwNSIgZm9udC1mYW1pbHk9Im1
vbm9zcGFjZSIgZm9udC1zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9IiM3YzNhZWQi
PlRJTUU8L3RleHQ+CiAgPHRleHQgeD0iNDgwIiB5PSIzMDUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc
2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzMzMyI+VGltZSAoSEg6TU06U1MpPC90ZX
h0PgogIDx0ZXh0IHg9IjQyNSIgeT0iMzI1IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXN
pemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0iIzdjM2FlZCI+RVBPQ0g8L3RleHQ+CiAg
PHRleHQgeD0iNDgwIiB5PSIzMjUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9ud
C1zaXplPSIxMCIgZmlsbD0iIzMzMyI+VW5peCB0aW1lc3RhbXA8L3RleHQ+CiAgPHRleHQgeD0iND
I1IiB5PSIzNDUiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTEiIGZvbnQtd2V
pZ2h0PSJib2xkIiBmaWxsPSIjN2MzYWVkIj5EVVJBVElPTjwvdGV4dD4KICA8dGV4dCB4PSI1MDAi
IHk9IjM0NSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBma
WxsPSIjMzMzIj5UaW1lIGR1cmF0aW9uPC90ZXh0PgoKICA8dGV4dCB4PSI2MjAiIHk9IjI2NSIgZm

```
9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzY0NzQ4YiI+MjAyNC0
wMS0xNVQuLi48L3RleHQ+CiAgPHRleHQgeD0iNjIwIiB5PSIyODUiIGZvbnQtZmFtaWx5PSJtb25v
c3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NDc0OGIiPjIwMjQtMDEtMTU8L3RleHQ+CiAgP
HRleHQgeD0iNjIwIiB5PSIzMDUiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMT
AiIGZpbGw9IiM2NDc0OGIiPjE0OjMwOjAwPC90ZXh0PgogIDx0ZXh0IHg9IjYyMCIgeT0iMzI1IiB
mb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNjQ3NDhiIj4xNzA1
MzI5NjAwPC90ZXh0PgogIDx0ZXh0IHg9IjYyMCIgeT0iMzQ1IiBmb250LWZhbWlseT0ibW9ub3NwY
WNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNjQ3NDhiIj4yaDMwbSwgMWQ8L3RleHQ+CgogIDwhLS
0gU3ludGF4IHRpcCAtLT4KICA8cmVjdCB4PSIzMCIgeT0iMzYwIiB3aWR0aD0iNzQwIiBoZWlnaHQ
9IjEyIiByeD0iNCIgZmlsbD0iI2UwZjJmZSIvPgogIDx0ZXh0IHg9IjQwIiB5PSIzNjQiIGZvbnQ
tZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiMzMzY5YTEiI
HRleHQtYW5jaG9yPSJtaWRkbGUiPlN5bnRheDogTUFUQ0hFUjpmaWVsZG5hbWUgKGUuZy4sIElOVD
pzdGF0dXNfY29kZSkgb3IgTUFUQ0hFUiAobm8gY2FwdHVyZSk8L3RleHQ+Cjwvc3ZnPgo=)



| Matcher | Description | Example Match |
|---------|-------------|---------------|
| `LD` | Line data (to delimiter) | Any text |
| `INT` | Integer | `42`, `-17` |
| `DOUBLE` | Decimal number | `3.14`, `-0.5` |
| `IPADDR` | IP address | `192.168.1.1` |
| `WORD` | Word characters | `hello123` |
| `TIMESTAMP` | Date/time | Various formats |
| `JSON` | JSON object/array | `{"key": "value"}` |

### OpenPipeline Parse Processor

```yaml
# Parse HTTP access logs
processors:
  - name: parse-http-logs
    type: dql
    source: content
    dql: |
      parse content, "IPADDR:client_ip SPACE LD SPACE '['
TIMESTAMP:request_time ']'
                     SPACE '\"' LD:method SPACE LD:path SPACE LD '\"'
                     SPACE INT:status SPACE INT:bytes"
```

```python
// Discover log patterns for parsing design
fetch logs, from: now() - 1h
| fieldsAdd content_preview = substring(content, from: 0, to: 100)
| summarize {count = count()}, by: {content_preview}
| sort count desc
| limit 20
```
```

```

```python
// Test parse pattern before configuring in OpenPipeline
fetch logs, from: now() - 1h
| filter contains(content, "GET") OR contains(content, "POST")
| parse content, "LD:method SPACE '/' LD:path SPACE INT:status_code"
| filter isNotNull(status_code)
| summarize {count = count()}, by: {method, status_code}
| sort count desc
| limit 15
```

```python
// Verify fields already extracted by OpenPipeline
fetch logs, from: now() - 1h
| limit 100
| summarize {
    has_loglevel = countIf(isNotNull(loglevel)),
    has_status = countIf(isNotNull(status)),
    has_trace_id = countIf(isNotNull(trace_id)),
    has_span_id = countIf(isNotNull(span_id))
  }
```

3. Metric Extraction from Logs

Extract metrics with dimensions from log data at ingestion time. This is powerful for:
- Creating SLIs from log patterns
- Building dashboards without log queries
- Enabling metric-based alerting

OpenPipeline Metric Extraction

```yaml
# Extract request duration metric from logs
processors:
  - name: extract-request-metric
    type: metric
    enabled: true
    condition: contains(content, "duration=")
    metricKey: log.request.duration
    dimensions:
      - service: k8s.namespace.name
      - method: extracted_method
      - status: extracted_status
    value: extracted_duration_ms
```
```

```

### Common Metric Extraction Patterns

| Log Pattern | Metric | Dimensions |
|-------------|--------|------------|
| Request completed | `log.request.count` | service, method, status |
| Response time | `log.response.duration` | endpoint, status_code |
| Error occurred | `log.error.count` | error_type, service |
| Queue depth | `log.queue.size` | queue_name |

```python
// Identify logs with numeric values for metric extraction
fetch logs, from: now() - 1h
| filter contains(content, "duration")
        OR contains(content, "latency")
        OR contains(content, "time=")
        OR contains(content, "ms")
| fieldsAdd content_preview = substring(content, from: 0, to: 120)
| summarize {count = count()}, by: {content_preview}
| sort count desc
| limit 15
```

```python
// Simulate metric extraction: count by dimensions
fetch logs, from: now() - 1h
| filter isNotNull(k8s.namespace.name)
| summarize {
    request_count = count(),
    error_count = countIf(loglevel == "ERROR")
  }, by: {k8s.namespace.name, k8s.workload.name}
| fieldsAdd error_rate = round((error_count * 100.0) / request_count,
decimals: 2)
| sort error_count desc
| limit 15
```

```python
// Preview: What dimensions would be valuable?
fetch logs, from: now() - 1h
| summarize {
    unique_namespaces = countDistinct(k8s.namespace.name),
    unique_workloads = countDistinct(k8s.workload.name),
    unique_pods = countDistinct(k8s.pod.name),
    unique_hosts = countDistinct(dt.entity.host),
    unique_sources = countDistinct(dt.openpipeline.source)
  }
```
```

```

## 4. Attribute Creation & Enrichment

Add **computed attributes** to logs for enhanced analysis and filtering.

### OpenPipeline Field Processor

```yaml
# Add computed attributes
processors:
  - name: enrich-environment
    type: fieldsAdd
    fields:
      - name: environment
        value: |
          if(contains(k8s.namespace.name, "prod"), "production",
          else: if(contains(k8s.namespace.name, "staging"), "staging",
          else: "development"))

      - name: severity_score
        value: |
          if(loglevel == "ERROR", 3,
          else: if(loglevel == "WARN", 2,
          else: 1))

      - name: team_owner
        value: |
          if(contains(k8s.namespace.name, "payment"), "platform-team",
          else: if(contains(k8s.namespace.name, "frontend"), "web-team",
          else: "unknown"))
```

Common Attribute Patterns

Attribute	Source	Use Case
`environment`	namespace name	Filter prod vs dev
`team_owner`	namespace/labels	Route alerts
`severity_score`	loglevel	Prioritization
`log_category`	content patterns	Classification

```python
// Preview attribute creation logic
fetch logs, from: now() - 1h
| filter isNotNull(k8s.namespace.name)
| fieldsAdd environment = if(contains(k8s.namespace.name, "prod"),
"production",
```
```

```
                           else: if(contains(k8s.namespace.name, "staging"),
"staging",
                           else: "development"))
| fieldsAdd severity_score = if(loglevel == "ERROR", 3,
                              else: if(loglevel == "WARN", 2,
                              else: 1))
| summarize {count = count()}, by: {environment, severity_score, loglevel}
| sort environment asc, severity_score desc
```

```python
// Categorize logs by content patterns
fetch logs, from: now() - 1h
| fieldsAdd log_category = if(contains(content, "Exception") OR
contains(content, "Error"), "exception",
                           else: if(contains(content, "request") OR
contains(content, "response"), "http",
                           else: if(contains(content, "database") OR
contains(content, "query"), "database",
                           else: if(contains(content, "auth") OR
contains(content, "login"), "security",
                           else: "general"))))
| summarize {count = count()}, by: {log_category, loglevel}
| sort count desc
```

```python
// Identify namespaces for team ownership mapping
fetch logs, from: now() - 1h
| filter isNotNull(k8s.namespace.name)
| summarize {log_count = count()}, by: {k8s.namespace.name}
| sort log_count desc
| limit 20
```

5. Event Generation from Logs

Create **business events** from specific log patterns. Events flow to Grail and can trigger workflows.

OpenPipeline Event Processor

```yaml
# Generate events from critical log patterns
processors:
  - name: generate-payment-events
    type: bizevents
    enabled: true
```

```
    condition: contains(content, "payment") AND contains(content,
"completed")
    eventType: com.example.payment.completed
    attributes:
      - payment_id: extracted_payment_id
      - amount: extracted_amount
      - currency: extracted_currency
      - customer_id: extracted_customer

  - name: generate-error-events
    type: bizevents
    enabled: true
    condition: loglevel == "ERROR" AND contains(content, "critical")
    eventType: com.example.critical.error
    attributes:
      - error_type: extracted_error_type
      - service: k8s.namespace.name
```

Event Use Cases

Log Pattern	Event Type	Purpose
Order completed	`order.completed`	Business analytics
User signup	`user.registered`	Funnel tracking
Deployment	`deployment.completed`	Change tracking
Critical error	`error.critical`	Incident trigger

```python
// Discover patterns suitable for event generation
fetch logs, from: now() - 1h
| filter contains(content, "completed")
        OR contains(content, "success")
        OR contains(content, "failed")
        OR contains(content, "created")
| fieldsAdd content_preview = substring(content, from: 0, to: 100)
| summarize {count = count()}, by: {content_preview}
| sort count desc
| limit 20
```

```python
// Check existing bizevents (if any generated)
fetch bizevents, from: now() - 24h
| summarize {count = count()}, by: {event.type}
| sort count desc
| limit 15
```

```python
// Preview: Logs that would become critical error events
fetch logs, from: now() - 1h
| filter loglevel == "ERROR"
| filter contains(content, "critical")
        OR contains(content, "fatal")
        OR contains(content, "severe")
| fieldsAdd content_preview = substring(content, from: 0, to: 100)
| fields timestamp, k8s.namespace.name, content_preview
| sort timestamp desc
| limit 20
```

6. Bucket Routing

Route logs to **appropriate buckets** based on content, source, or computed attributes.

Why Bucket Routing Matters

Bucket	Logs	Retention	Cost Impact
`default_logs`	Standard	35 days	Baseline
`debug_logs`	DEBUG/TRACE	7 days	80% savings
`audit_logs`	Security/compliance	365 days	Compliance
`error_logs`	Errors only	90 days	Investigation

OpenPipeline Route Processor

```yaml
# Route logs to appropriate buckets
processors:
  - name: route-debug-logs
    type: route
    condition: loglevel == "DEBUG" OR loglevel == "TRACE"
    bucket: debug_logs

  - name: route-audit-logs
    type: route
    condition: contains(content, "audit") OR contains(content, "security")
    bucket: audit_logs

  - name: route-error-logs
    type: route
    condition: loglevel == "ERROR" OR loglevel == "FATAL"
    bucket: error_logs
```

```python
// Current bucket distribution
fetch logs, from: now() - 1h
| summarize {count = count()}, by: {dt.system.bucket}
| sort count desc
```

```python
// Preview routing decisions
fetch logs, from: now() - 1h
| fieldsAdd target_bucket = if(loglevel == "DEBUG" OR loglevel == "TRACE",
"debug_logs",
                           else: if(loglevel == "ERROR" OR loglevel ==
"FATAL", "error_logs",
                           else: if(contains(content, "audit") OR
contains(content, "security"), "audit_logs",
                           else: "default_logs")))
| summarize {count = count()}, by: {target_bucket, loglevel}
| sort count desc
```

```python
// Estimate storage savings from routing
fetch logs, from: now() - 24h
| fieldsAdd content_bytes = stringLength(content)
| summarize {
    total_logs = count(),
    total_mb = sum(content_bytes) / 1048576.0,
    debug_logs = countIf(loglevel == "DEBUG" OR loglevel == "TRACE"),
    debug_mb = sum(if(loglevel == "DEBUG" OR loglevel == "TRACE",
content_bytes, else: 0)) / 1048576.0
  }
| fieldsAdd debug_pct = round((debug_logs * 100.0) / total_logs, decimals: 1)
| fieldsAdd savings_7d_vs_35d = round(debug_mb * 0.8, decimals: 2)
```

7. Filtering & Sampling

Reduce log volume by **filtering** or **sampling** at ingestion.

OpenPipeline Filter Processor

```yaml
# Drop noisy, low-value logs
processors:
  - name: drop-health-checks
    type: filter
```

```
    condition: contains(content, "health") AND loglevel == "INFO"
    action: drop

  - name: drop-heartbeats
    type: filter
    condition: contains(content, "heartbeat") OR contains(content,
"keepalive")
    action: drop

  - name: sample-debug-logs
    type: sample
    condition: loglevel == "DEBUG"
    rate: 0.1  # Keep only 10% of DEBUG logs
```

```python
// Identify candidates for filtering (high volume, low value)
fetch logs, from: now() - 1h
| filter contains(content, "health")
        OR contains(content, "heartbeat")
        OR contains(content, "alive")
        OR contains(content, "ready")
| summarize {count = count()}, by: {k8s.namespace.name, loglevel}
| sort count desc
| limit 15
```

```python
// Calculate potential savings from filtering health checks
fetch logs, from: now() - 24h
| summarize {
    total_logs = count(),
    health_logs = countIf(contains(content, "health") OR contains(content,
"heartbeat")),
    debug_logs = countIf(loglevel == "DEBUG")
  }
| fieldsAdd health_pct = round((health_logs * 100.0) / total_logs, decimals:
1)
| fieldsAdd debug_pct = round((debug_logs * 100.0) / total_logs, decimals: 1)
| fieldsAdd potential_reduction = round(((health_logs + debug_logs * 0.9) *
100.0) / total_logs, decimals: 1)
```

8. Complete Pipeline Example

Here's a comprehensive OpenPipeline configuration:

```yaml
```

```
# Complete OpenPipeline configuration for logs
name: production-log-pipeline
enabled: true

processors:
  # 1. FILTER - Remove unwanted logs first
  - name: drop-health-checks
    type: filter
    condition: contains(content, "health") AND loglevel == "INFO"
    action: drop

  # 2. PARSE - Extract structured fields
  - name: parse-http-logs
    type: dql
    condition: contains(content, "HTTP")
    dql: parse content, "LD:method SPACE '/' LD:path SPACE INT:status SPACE
INT:duration_ms"

  # 3. ENRICH - Add computed attributes
  - name: add-environment
    type: fieldsAdd
    fields:
      - name: environment
        value: if(contains(k8s.namespace.name, "prod"), "production", else:
"non-prod")

  # 4. MASK - Protect sensitive data
  - name: mask-emails
    type: mask
    field: content
    pattern: "[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}"
    replacement: "[EMAIL-MASKED]"

  # 5. EXTRACT METRICS - Create dimensional metrics
  - name: extract-request-count
    type: metric
    metricKey: log.http.requests
    dimensions:
      - namespace: k8s.namespace.name
      - status: status

  # 6. GENERATE EVENTS - Create business events
  - name: generate-error-events
    type: bizevents
    condition: loglevel == "ERROR"
    eventType: com.app.error

  # 7. ROUTE - Send to appropriate bucket
```

```
  - name: route-debug
    type: route
    condition: loglevel == "DEBUG"
    bucket: debug_logs
```

```python
// Verify current pipeline processing
fetch logs, from: now() - 1h
| summarize {
    total_logs = count(),
    with_pipeline = countIf(isNotNull(dt.openpipeline.pipelines)),
    unique_pipelines = countDistinct(dt.openpipeline.pipelines)
  }, by: {dt.openpipeline.source}
| sort total_logs desc
```

```python
// Pipeline processing summary
fetch logs, from: now() - 1h
| summarize {count = count()}, by: {dt.openpipeline.pipelines,
dt.system.bucket}
| sort count desc
```

📝 Summary

In this notebook, you learned:

✅ **Parsing** - Extract structured fields at ingestion
✅ **Metrics** - Create dimensional metrics from logs
✅ **Attributes** - Add computed fields for analysis
✅ **Events** - Generate business events from patterns
✅ **Routing** - Direct logs to appropriate buckets
✅ **Filtering** - Drop/sample low-value logs

Key Takeaway

> **Process at ingestion, not at query time.** OpenPipeline processing is fundamental to cost optimization, data quality, and operational efficiency.

➡️ Next Steps

Continue to **OPLOGS-04: Buckets & Data Governance** to learn about storage

```
management and retention policies.

---

## 📚 References

- [OpenPipeline Overview]
(https://docs.dynatrace.com/docs/platform/openpipeline)
- [OpenPipeline Processors]
(https://docs.dynatrace.com/docs/platform/openpipeline/configuration/processo
rs)
- [Metric Extraction]
(https://docs.dynatrace.com/docs/platform/openpipeline/use-cases/log-
processing/extract-metrics)
- [Event Generation]
(https://docs.dynatrace.com/docs/platform/openpipeline/use-cases/log-
processing/create-business-events)
- [DPL Pattern Language]
(https://docs.dynatrace.com/docs/platform/grail/dynatrace-pattern-language)
```