🗺️ Topology & Entity Context

> **Series:** OPLOGS | **Notebook:** 6 of 8 | **Created:** December 2025

Leveraging Entity Relationships in Log Analysis

This notebook explores how Dynatrace enriches logs with entity context
(hosts, processes, services, Kubernetes) for topology-aware analysis.

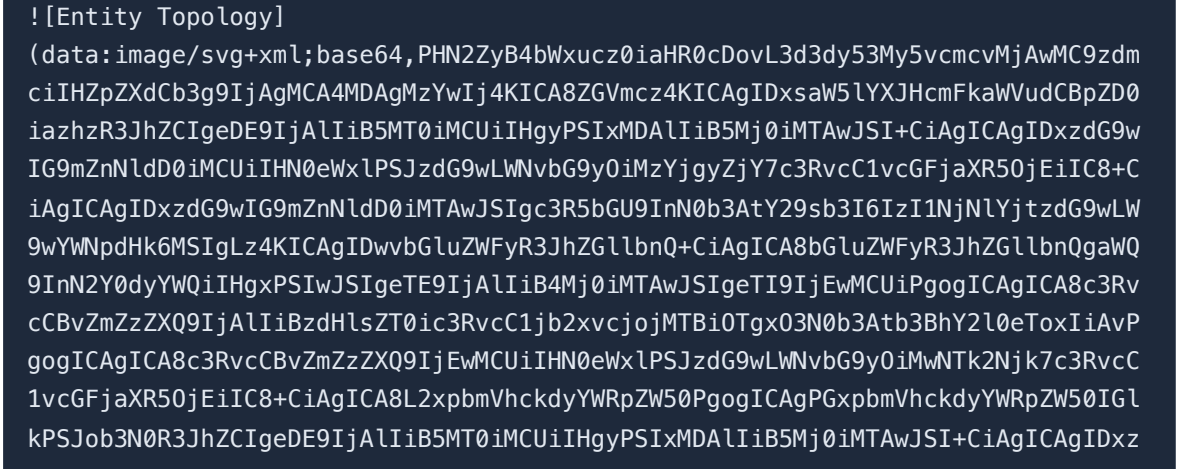
Table of Contents

Prerequisites

- ✅ Access to a Dynatrace environment with log data
- ✅ Completed OPLOGS-01 through OPLOGS-05
- ✅ Understanding of Dynatrace entity model (helpful)

1. Entity Types Overview

Dynatrace automatically enriches logs with entity context:



dG9wIG9mZnNldD0iMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiNmNTllMGI7c3RvcC1vcGFjaXR5OjEiI
C8+CiAgICAgIDxzdG9wIG9mZnNldD0iMTAwJSIgc3R5bGU9InN0b3AtY29sb3I6I2Q5NzcwNjtzdG
9wLW9wYWNpdHk6MSIgLz4KICAgIDwvbGluZWFyR3JhZGllbnQ+CiAgICA8bGluZWFyR3JhZGllbnQ
gaWQ9ImxvZ0dyYWQiIHgxPSIwJSIgeTE9IjAlIiB4Mj0iMTAwJSIgeTI9IjEwMCUiPgogICAgICA8
c3RvcCBvZmZzZXQ9IjAlIiBzdHlsZT0ic3RvcC1jb2xvcjojOGI1Y2Y2O3N0b3Atb3BhY2l0eToxI
iAvPgogICAgICA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiM3YzNhZWQ7c3
RvcC1vcGFjaXR5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50PgogICAgPGZpbHRlciBpZD0idG9
wb1NoYWRvdyI+CiAgICAgIDxmZURyb3BTaGFkb3cgZHg9IjEiIGR5PSIxIiBzdGREZXZpYXRpb249
IjIiIGZsb29kLW9wYWNpdHk9IjAuMTUiLz4KICAgIDwvZmlsdGVyPgogICAgPG1hcmtlciBpZD0id
G9wb0Fycm93IiBtYXJrZXJXaWR0aD0iMTAiIG1hcmtlckhlaWdodD0iNyIgcmVmWD0iOSIgcmVmWT
0iMy41IiBvcmllbnQ9ImF1dG8iPgogICAgICA8cG9seWdvbiBwb2ludHM9IjAgMCwgMTAgMy41LCA
wIDciIGZpbGw9IiM2NDc0OGIiLz4KICAgIDwvbWFya2VyPgogIDwvZGVmcz4KCiAgPCEtLSBCYWNr
Z3JvdW5kIC0tPgogIDxyZWN0IHdpZHRoPSI4MDAiIGhlaWdodD0iMzYwIiBmaWxsPSIjZjhmOWZhI
iByeD0iMTAiLz4KCiAgPCEtLSBUaXRsZSAtLT4KICA8dGV4dCB4PSI0MDAiIHk9IjI4IiBmb250LW
ZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTgiIGZvbnQtd2VpZ2h0PSJib2x
kIiBmaWxsPSIjMzMzIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5FbnRpdHkgVG9wb2xvZ3kgQ29udGV4
dCBmb3IgTG9nczwvdGV4dD4KICA8dGV4dCB4PSI0MDAiIHk9IjQ4IiBmb250LWZhbWlseT0iQXJpY
WwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZpbGw9IiM2NjYiIHRleHQtYW5jaG9yPSJtaW
RkbGUiPkxvZ3MgYXV0b21hdGljYWxseSBsaW5rZWQgdG8gZW50aXRpZXMgdmlhIGR0LmVudGl0eS4
qIGZpZWxkczwvdGV4dD4KCiAgPCEtLSBLdWJlcm5ldGVzIEhpZXJhcmNoeSAtLT4KICA8cmVjdCB4
PSIzMCIgeT0iNzAiIHdpZHRoPSIzODAiIGhlaWdodD0iMTk1IiByeD0iMTAiIGZpbGw9IiNmZmYiI
HN0cm9rZT0iI2UyZThmMCIgc3Ryb2tlLXdpZHRoPSIyIi8+CiAgPHRleHQgeD0iMjIwIiB5PSI5NS
IgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEyIiBmb250LXdlaWd
odD0iYm9sZCIgZmlsbD0iIzMzMyIgdGV4dC1hbmNob3I9Im1pZGRsZSI+S3ViZXJuZXRlcyBFbnRp
dHkgSGllcmFyY2h5PC90ZXh0PgoKICA8IS0tIENsdXN0ZXIgLS0+CiAgPHJlY3QgeD0iNTAiIHk9I
jExMCIgd2lkdGg9IjM0MCIgaGVpZ2h0PSIxNDUiIHJ4PSI4IiBmaWxsPSIjZGJlYWZlIiBzdHJva2
U9IiMzYjgyZjYiIHN0cm9rZS13aWR0aD0iMSIvPgogIDx0ZXh0IHg9IjcwIiB5PSIxMzAiIGZvbnQ
tZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJv
bGQiIGZpbGw9IiMxZTQwYWYiPktVQkVSTkVURVNfQ0xVU1RFUjwvdGV4dD4KICA8dGV4dCB4PSI3M
CIgeT0iMTQ1IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjM2
I4MmY2Ij5kdC5lbnRpdHkua3ViZXJuZXRlc19jbHVzdGVyPC90ZXh0PgoKICA8IS0tIE5hbWVzcGF
jZSAtLT4KICA8cmVjdCB4PSI3MCIgeT0iMTU1IiB3aWR0aD0iMzAwIiBoZWlnaHQ9IjkwIiByeD0i
NiIgZmlsbD0iI2VjZmRmNSIgc3Ryb2tlPSIjMTBiOTgxIiBzdHJva2Utd2lkdGg9IjEiLz4KICA8d
GV4dCB4PSI5MCIgeT0iMTc1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2
l6ZT0iMTEiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSIjMDQ3ODU3Ij5DTE9VRF9BUFBMSUNBVEl
PTl9OQU1FU1BBQ0U8L3RleHQ+CiAgPHRleHQgeD0iOTAiIHk9IjE5MCIgZm9udC1mYW1pbHk9Im1v
bm9zcGFjZSIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzA1OTY2OSI+ZHQuZW50aXR5LmNsb3VkX2Fwc
GxpY2F0aW9uX25hbWVzcGFjZTwvdGV4dD4KCiAgPCEtLSBQb2QgLS0+CiAgPHJlY3QgeD0iOTAiIH
k9IjIwMCIgd2lkdGg9IjEzMCIgaGVpZ2h0PSIzNSIgcng9IjQiIGZpbGw9InVybCgjazhzR3JhZCk
iIGZpbHRlcj0idXJsKCN0b3BvU2hhZG93KSIvPgogIDx0ZXh0IHg9IjE1NSIgeT0iMjE1IiBmb250
LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZvbnQtd2VpZ2h0PSJib
2xkIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+UE9EPC90ZXh0PgogIDx0ZXh0IH
g9IjE1NSIgeT0iMjI4IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0
iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+ZHQu
ZW50aXR5LnBvZDwvdGV4dD4KCiAgPCEtLSBDb250YWluZXIgLS0+CiAgPHJlY3QgeD0iMjMwIiB5P
SIyMDAiIHdpZHRoPSIxMzAiIGhlaWdodD0iMzUiIHJ4PSI0IiBmaWxsPSJ1cmwoI2s4c0dyYWQpIi
BmaWx0ZXI9InVybCgjdG9wb1NoYWRvdykiLz4KICA8dGV4dCB4PSIyOTUiIHk9IjIxNSIgZm9udC1
mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmb250LXdlaWdodD0iYm9s

ZCIgZmlsbD0id2hpdGUiIHRleHQtYW5jaG9yPSJtaWRkbGUiPkNPTlRBSU5FUjwvdGV4dD4KICA8d
GV4dCB4PSIyOTUiIHk9IjIyOCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LX
NpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGU
iPmR0LmVudGl0eS5jb250YWluZXI8L3RleHQ+CgogIDwhLS0gU2VydmljZS9Ib3N0IEVudGl0aWVz
IC0tPgogIDxyZWN0IHg9IjQzMCIgeT0iNzAiIHdpZHRoPSIzNDAiIGhlaWdodD0iMTk1IiByeD0iM
TAiIGZpbGw9IiNmZmYiIHN0cm9rZT0iI2UyZThmMCIgc3Ryb2tlLXdpZHRoPSIyIi8+CiAgPHRleH
QgeD0iNjAwIiB5PSI5NSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU
9IjEyIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0iIzMzMyIgdGV4dC1hbmNob3I9Im1pZGRsZSI+
U2VydmljZSAmIEhvc3QgRW50aXRpZXM8L3RleHQ+CgogIDxyZWN0IHg9IjQ1MCIgeT0iMTEwIiB3a
WR0aD0iMTQ1IiBoZWlnaHQ9IjY1IiByeD0iNiIgZmlsbD0idXJsKCNzdmNHcmFkKSIgZmlsdGVyPS
J1cmwoI3RvcG9TaGFkb3cpIi8+CiAgPHRleHQgeD0iNTIyIiB5PSIxMzUiIGZvbnQtZmFtaWx5PSJ
BcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9
IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5TRVJWSUNFPC90ZXh0PgogIDx0ZXh0IHg9IjUyM
iIgeT0iMTU1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIG
ZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+ZHQuZW50aXR
5LnNlcnZpY2U8L3RleHQ+CiAgPHRleHQgeD0iNTIyIiB5PSIxNjgiIGZvbnQtZmFtaWx5PSJBcmlh
bCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0icmdiYSgyNTUsMjU1LDI1NSwwLjkpI
iB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5EZXRlY3RlZCBzZXJ2aWNlczwvdGV4dD4KCiAgPHJlY3QgeD
0iNjA1IiB5PSIxMTAiIHdpZHRoPSIxNDUiIGhlaWdodD0iNjUiIHJ4PSI2IiBmaWxsPSJ1cmwoI2h
vc3RHcmFkKSIgZmlsdGVyPSJ1cmwoI3RvcG9TaGFkb3cpIi8+CiAgPHRleHQgeD0iNjc3IiB5PSIx
MzUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMSIgZm9udC13Z
WlnaHQ9ImJvbGQiIGZpbGw9IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5IT1NUPC90ZXh0Pg
ogIDx0ZXh0IHg9IjY3NyIgeT0iMTU1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZ
vbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1p
ZGRsZSI+ZHQuZW50aXR5Lmhvc3Q8L3RleHQ+CiAgPHRleHQgeD0iNjc3IiB5PSIxNjgiIGZvbnQtZ
mFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0icmdiYSgyNTUsMj
U1LDI1NSwwLjkpIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5QaHlzaWNhbC9WTSBob3N0czwvdGV4dD4
KCiAgPHJlY3QgeD0iNDUwIiB5PSIxODUiIHdpZHRoPSIxNDUiIGhlaWdodD0iNjUiIHJ4PSI2IiBm
aWxsPSIjZmVmM2M3IiBzdHJva2U9IiNmNTllMGIiIHN0cm9rZS13aWR0aD0iMSIvPgogIDx0ZXh0I
Hg9IjUyMiIgeT0iMjEwIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT
0iMTEiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSIjOTI0MDBlIiB0ZXh0LWFuY2hvcj0ibWlkZGx
lIj5QUk9DRVNTX0dST1VQPC90ZXh0PgogIDx0ZXh0IHg9IjUyMiIgeT0iMjI4IiBmb250LWZhbWls
eT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM3ODM1MGYiIHRleHQtY
W5jaG9yPSJtaWRkbGUiPmR0LmVudGl0eS5wcm9jZXNzX2dyb3VwPC90ZXh0PgogIDx0ZXh0IHg9Ij
UyMiIgeT0iMjQzIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTA
iIGZpbGw9IiM3ODM1MGYiIHRleHQtYW5jaG9yPSJtaWRkbGUiPlJ1bm5pbmcgcHJvY2Vzc2VzPC90
ZXh0PgoKICA8cmVjdCB4PSI2MDUiIHk9IjE4NSIgd2lkdGg9IjE0NSIgaGVpZ2h0PSI2NSIgcng9I
jYiIGZpbGw9IiNmY2U3ZjMiIHN0cm9rZT0iI2VjNDg5OSIgc3Ryb2tlLXdpZHRoPSIxIi8+CiAgPH
RleHQgeD0iNjc3IiB5PSIyMTAiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1
zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9IiM5ZDE3NGQiIHRleHQtYW5jaG9yPSJt
aWRkbGUiPkFQUExJQ0FUSU9OPC90ZXh0PgogIDx0ZXh0IHg9IjY3NyIgeT0iMjI4IiBmb250LWZhb
WlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM5ZDE3NGQiIHRleH
QtYW5jaG9yPSJtaWRkbGUiPmR0LmVudGl0eS5hcHBsaWNhdGlvbjwvdGV4dD4KICA8dGV4dCB4PSI
2NzciIHk9IjI0MyIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEw
IiBmaWxsPSIjOWQxNzRkIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5SVU0gYXBwbGljYXRpb25zPC90Z
Xh0PgoKICA8IS0tIExvZyBSZWNvcmQgLS0+CiAgPHJlY3QgeD0iMzAiIHk9IjI4MCIgd2lkdGg9Ij
c0MCIgaGVpZ2h0PSI2NSIgcng9IjEwIiBmaWxsPSIjZmZmIiBzdHJva2U9IiNlMmU4ZjAiIHN0cm9
rZS13aWR0aD0iMiIvPgogIDxyZWN0IHg9IjMwIiB5PSIyODAiIHdpZHRoPSI3NDAiIGhlaWdodD0i

```
MjUiIHJ4PSIxMCIgZmlsbD0idXJsKCNsb2dHcmFkKSIvPgogIDx0ZXh0IHg9IjQwMCIgeT0iMjk4I
iBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZvbnQtd2VpZ2
h0PSJib2xkIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+TG9nIFJlY29yZCB3aXR
oIEVudGl0eSBDb250ZXh0PC90ZXh0PgoKICA8dGV4dCB4PSI1MCIgeT0iMzI1IiBmb250LWZhbWls
eT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjMzMzIj5mZXRjaCBsb2dzIHwgZmllb
GRzIHRpbWVzdGFtcCwgY29udGVudCwgZHQuZW50aXR5Lmhvc3QsIGR0LmVudGl0eS5zZXJ2aWNlLC
BkdC5lbnRpdHkua3ViZXJuZXRlc19jbHVzdGVyLCBkdC5lbnRpdHkucG9kPC90ZXh0PgoKICA8IS0
tIEFycm93cyBzaG93aW5nIHJlbGF0aW1uc2hpcCAtLT4KICA8cGF0aCBkPSJNMTU1LDIzNSBRMTU1
LDI3MCAyNTAsMjc1IiBzdHJva2U9IiMzYjgyZjYiIHN0cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZ
SIgc3Ryb2tlLWRhc2hhcnJheT0iNCwyIiBtYXJrZXItZW5kPSJ1cmwoI3RvcG9BcnJvdykiLz4KICA
8cGF0aCBkPSJNNTIyLDE3NSBRMTIyLDI1MCA0MDAsMjc1IiBzdHJva2U9IiMxMGI5ODEiIHN0cm9r
ZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgc3Ryb2tlLWRhc2hhcnJheT0iNCwyIiBtYXJrZXItZW5k
PSJ1cmwoI3RvcG9BcnJvdykiLz4KICA8cGF0aCBkPSJNNjc3LDE3NSBRNjc3LDI1MCA1NTAsMjc1I
iBzdHJva2U9IiNmNTllMGIiIHN0cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgc3Ryb2tlLWRhc2
hhcnJheT0iNCwyIiBtYXJrZXItZW5kPSJ1cmwoI3RvcG9BcnJvdykiLz4KPC9zdmc+Cg==)



| Entity Field | Description | Example |
|--------------|-------------|----------|
| `dt.entity.host` | Host entity ID | `HOST-ABC123` |
| `dt.entity.process_group` | Process group ID | `PROCESS_GROUP-XYZ789` |
| `dt.entity.process_group_instance` | PGI ID | `PROCESS_GROUP_INSTANCE-
DEF456` |
| `dt.entity.service` | Service entity ID | `SERVICE-QRS012` |
| `dt.entity.kubernetes_cluster` | K8s cluster ID | `KUBERNETES_CLUSTER-
TUV345` |

### Kubernetes Context Fields

| Field | Description |
|-------|-------------|
| `k8s.namespace.name` | Kubernetes namespace |
| `k8s.pod.name` | Pod name |
| `k8s.pod.uid` | Pod unique identifier |
| `k8s.container.name` | Container name |
| `k8s.cluster.name` | Cluster name |
| `k8s.deployment.name` | Deployment name |
| `k8s.workload.name` | Workload name |
| `k8s.workload.kind` | Workload type (Deployment, StatefulSet, etc.) |

```python
// Discover available entity types in your logs
fetch logs, from: now() - 1h
| summarize {
    total_logs = count(),
    with_host = countIf(isNotNull(dt.entity.host)),
    with_process_group = countIf(isNotNull(dt.entity.process_group)),
```
```

````
    with_service = countIf(isNotNull(dt.entity.service)),
    with_k8s_cluster = countIf(isNotNull(dt.entity.kubernetes_cluster)),
    with_k8s_namespace = countIf(isNotNull(k8s.namespace.name))
  }
```

## 2. Host Topology

Analyze logs by host to understand infrastructure patterns.

```python
// Log volume by host
fetch logs, from: now() - 1h
| filter isNotNull(dt.entity.host)
| summarize {log_count = count()}, by: {dt.entity.host}
| sort log_count desc
| limit 15
```

```python
// Error distribution by host
fetch logs, from: now() - 1h
| filter isNotNull(dt.entity.host)
| summarize {
    total = count(),
    errors = countIf(loglevel == "ERROR" OR loglevel == "SEVERE")
  }, by: {dt.entity.host}
| fieldsAdd error_rate = (errors * 100.0) / total
| sort errors desc
| limit 15
```

```python
// Host with log.source breakdown
fetch logs, from: now() - 1h
| filter isNotNull(dt.entity.host)
| summarize {count = count()}, by: {dt.entity.host, log.source}
| sort count desc
| limit 20
```

## 3. Process Group Topology

Process groups represent logical application components across hosts.

```python
// Logs by process group
fetch logs, from: now() - 1h
````

```
| filter isNotNull(dt.entity.process_group)
| summarize {log_count = count()}, by: {dt.entity.process_group}
| sort log_count desc
| limit 15
```

```python
// Process group error analysis
fetch logs, from: now() - 1h
| filter isNotNull(dt.entity.process_group)
| filter loglevel == "ERROR"
| fieldsAdd content_preview = substring(content, from: 0, to: 80)
| summarize {error_count = count()}, by: {dt.entity.process_group,
content_preview}
| sort error_count desc
| limit 20
```

```python
// Process group to host mapping
fetch logs, from: now() - 1h
| filter isNotNull(dt.entity.process_group) AND isNotNull(dt.entity.host)
| summarize {count = count()}, by: {dt.entity.process_group, dt.entity.host}
| sort count desc
| limit 20
```

4. Kubernetes Topology

OpenPipeline enriches container logs with rich Kubernetes context.

```python
// Logs by Kubernetes namespace
fetch logs, from: now() - 1h
| filter isNotNull(k8s.namespace.name)
| summarize {log_count = count()}, by: {k8s.namespace.name}
| sort log_count desc
| limit 15
```

```python
// Kubernetes namespace with error breakdown
fetch logs, from: now() - 1h
| filter isNotNull(k8s.namespace.name)
| summarize {
    total = count(),
    errors = countIf(loglevel == "ERROR" OR loglevel == "WARN")
  }, by: {k8s.namespace.name}
```

```
| fieldsAdd error_percentage = round((errors * 100.0) / total, decimals: 2)
| sort errors desc
| limit 10
```

```python
// Pod-level analysis
fetch logs, from: now() - 1h
| filter isNotNull(k8s.pod.name)
| summarize {
    log_count = count(),
    error_count = countIf(loglevel == "ERROR")
  }, by: {k8s.namespace.name, k8s.pod.name}
| sort error_count desc
| limit 20
```

```python
// Workload analysis (Deployments, StatefulSets, etc.)
fetch logs, from: now() - 1h
| filter isNotNull(k8s.workload.name)
| summarize {
    log_count = count(),
    unique_pods = countDistinct(k8s.pod.name)
  }, by: {k8s.namespace.name, k8s.workload.kind, k8s.workload.name}
| sort log_count desc
| limit 15
```

```python
// Container-level detail
fetch logs, from: now() - 1h
| filter isNotNull(k8s.container.name)
| summarize {log_count = count()}, by: {k8s.namespace.name, k8s.pod.name,
k8s.container.name}
| sort log_count desc
| limit 20
```

5. Service Mapping

Connect logs to Dynatrace-detected services for full observability.

```python
// Logs by service entity
fetch logs, from: now() - 1h
| filter isNotNull(dt.entity.service)
| summarize {log_count = count()}, by: {dt.entity.service}
```

```
| sort log_count desc
| limit 15
```

```python
// Service error rates from logs
fetch logs, from: now() - 1h
| filter isNotNull(dt.entity.service)
| summarize {
    total = count(),
    errors = countIf(loglevel == "ERROR")
  }, by: {dt.entity.service}
| fieldsAdd error_rate = round((errors * 100.0) / total, decimals: 2)
| sort error_rate desc
| limit 15
```

```python
// Service to process group relationship
fetch logs, from: now() - 1h
| filter isNotNull(dt.entity.service) AND isNotNull(dt.entity.process_group)
| summarize {count = count()}, by: {dt.entity.service,
dt.entity.process_group}
| sort count desc
| limit 20
```

6. Cross-Entity Correlation

Use entity context to correlate logs across the topology.

```python
// Full topology view: Cluster > Namespace > Pod > Container
fetch logs, from: now() - 1h
| filter isNotNull(k8s.cluster.name)
| summarize {log_count = count()}, by: {
    k8s.cluster.name,
    k8s.namespace.name,
    k8s.workload.name,
    k8s.pod.name
  }
| sort log_count desc
| limit 25
```

```python
// Entity coverage report
fetch logs, from: now() - 1h
```

```
| summarize {
    total_logs = count(),
    host_coverage = round((countIf(isNotNull(dt.entity.host)) * 100.0) /
count(), decimals: 1),
    pg_coverage = round((countIf(isNotNull(dt.entity.process_group)) * 100.0)
/ count(), decimals: 1),
    service_coverage = round((countIf(isNotNull(dt.entity.service)) * 100.0)
/ count(), decimals: 1),
    k8s_coverage = round((countIf(isNotNull(k8s.namespace.name)) * 100.0) /
count(), decimals: 1)
  }
```

```python
// Logs without entity context (potential configuration issue)
fetch logs, from: now() - 1h
| filter isNull(dt.entity.host) AND isNull(dt.entity.process_group)
| summarize {orphan_count = count()}, by: {dt.openpipeline.source}
| sort orphan_count desc
```

```python
// Trace correlation: Logs with trace context
fetch logs, from: now() - 1h
| filter isNotNull(trace_id) OR isNotNull(span_id)
| summarize {
    logs_with_trace = count(),
    unique_traces = countDistinct(trace_id)
  }, by: {k8s.namespace.name}
| sort logs_with_trace desc
| limit 10
```

7. Using Entity IDs for Lookups

Entity IDs enable cross-data-type correlation.

```python
// Get distinct entity IDs for a namespace
fetch logs, from: now() - 1h
| filter k8s.namespace.name == "hipstershop"
| summarize {
    unique_hosts = collectDistinct(dt.entity.host),
    unique_pgs = collectDistinct(dt.entity.process_group),
    unique_services = collectDistinct(dt.entity.service)
  }
```

```python
// Find logs for a specific entity (replace with actual entity ID)
// fetch logs, from: now() - 1h
// | filter dt.entity.host == "HOST-XXXXXX"
// | summarize {count = count()}, by: {loglevel}

// Discovery query to find entity IDs
fetch logs, from: now() - 1h
| filter isNotNull(dt.entity.host)
| summarize {sample = takeFirst(dt.entity.host)}, by: {k8s.namespace.name}
| limit 10
```

8. Topology-Based Alerting Patterns

Use entity context to create meaningful alert conditions.

```python
// Alert pattern: Errors per namespace (for threshold alerting)
fetch logs, from: now() - 15m
| filter loglevel == "ERROR"
| summarize {error_count = count()}, by: {k8s.namespace.name}
| filter error_count > 10
| sort error_count desc
```

```python
// Alert pattern: Hosts with high error rate
fetch logs, from: now() - 15m
| filter isNotNull(dt.entity.host)
| summarize {
    total = count(),
    errors = countIf(loglevel == "ERROR")
  }, by: {dt.entity.host}
| filter total > 100  // Minimum sample size
| fieldsAdd error_rate = (errors * 100.0) / total
| filter error_rate > 5  // Alert if >5% errors
| sort error_rate desc
```

```python
// Alert pattern: Pod restarts (look for startup patterns)
fetch logs, from: now() - 1h
| filter contains(content, "started") OR contains(content, "initializing")
| filter isNotNull(k8s.pod.name)
| summarize {startup_count = count()}, by: {k8s.namespace.name, k8s.pod.name}
| filter startup_count > 3  // More than 3 starts in 1h = potential crash
loop
```

```
| sort startup_count desc
```

📝 Summary

In this notebook, you learned:

✅ **Entity types** - HOST, PROCESS_GROUP, SERVICE, KUBERNETES_CLUSTER
✅ **Host topology** - Log volume and errors by host
✅ **Process groups** - Application component analysis
✅ **Kubernetes context** - Namespace, pod, workload, container
✅ **Service mapping** - Connecting logs to detected services
✅ **Cross-entity correlation** - Full topology views
✅ **Alerting patterns** - Threshold-based topology alerts

➡️ Next Steps

Continue to **OPLOGS-07: Analytics & Dashboards** for aggregation and visualization patterns.

📚 References

- [Dynatrace Entity Model](https://docs.dynatrace.com/docs/platform/grail/dynatrace-query-language/dql-guide/dql-entities)
- [Kubernetes Monitoring](https://docs.dynatrace.com/docs/shortlink/kubernetes-monitoring)
- [Log Enrichment](https://docs.dynatrace.com/docs/observe-and-explore/logs/log-management-and-analytics/lma-log-enrichment)