🔄 Migration to OpenPipeline

> **Series:** OPLOGS | **Notebook:** 2 of 8 | **Created:** December 2025

Planning and Executing Your Log Migration

This notebook guides you through assessing your current log environment and planning migration to OpenPipeline v2.0.

Table of Contents

Prerequisites

- ✅ Access to a Dynatrace environment with log data
- ✅ Completed OPLOGS-01 fundamentals
- ✅ Knowledge of your current log sources and volumes

1. Migration Assessment

Before migrating, you need to understand your current log landscape.

Key Questions to Answer:

1. **What log sources do I have?** (OneAgent, API, OTLP)
2. **What is my log volume?** (records/hour, GB/day)
3. **Which logs need parsing?** (unstructured content)
4. **What sensitive data exists?** (PII, credentials)
5. **What can be dropped?** (debug, health checks)

```python
// Assessment Query 1: Log sources inventory
fetch logs, from: now() - 24h
| summarize {
    log_count = count(),
    unique_hosts = countDistinct(dt.entity.host)
```

```
  }, by: {dt.openpipeline.source, log.source}
| sort log_count desc
| limit 20
```

```python
// Assessment Query 2: Log volume by hour (capacity planning)
fetch logs, from: now() - 24h
| makeTimeseries {log_count = count()}, interval: 1h
```

```python
// Assessment Query 3: Log level distribution
fetch logs, from: now() - 24h
| summarize {
    total = count(),
    errors = countIf(loglevel == "ERROR" OR status == "ERROR"),
    warnings = countIf(loglevel == "WARN" OR status == "WARN"),
    info = countIf(loglevel == "INFO" OR status == "INFO"),
    debug = countIf(loglevel == "DEBUG" OR status == "DEBUG"),
    none = countIf(loglevel == "NONE" OR status == "NONE")
  }
| fieldsAdd debug_pct = round((toDouble(debug) / toDouble(total)) * 100,
decimals: 1)
| fieldsAdd none_pct = round((toDouble(none) / toDouble(total)) * 100,
decimals: 1)
```

```python
// Assessment Query 4: Identify logs that need parsing (loglevel = NONE)
fetch logs, from: now() - 1h
| filter loglevel == "NONE" OR status == "NONE"
| fieldsAdd content_preview = substring(content, from: 0, to: 80)
| summarize {count = count()}, by: {content_preview, dt.openpipeline.source}
| sort count desc
| limit 15
```

```python
// Assessment Query 5: Potential drop candidates (health checks, heartbeats)
fetch logs, from: now() - 24h
| summarize {
    total = count(),
    health_checks = countIf(matchesPhrase(content, "health") OR
matchesPhrase(content, "healthcheck")),
    heartbeats = countIf(matchesPhrase(content, "heartbeat")),
    debug_logs = countIf(loglevel == "DEBUG" OR status == "DEBUG")
  }
```

```
| fieldsAdd droppable = health_checks + heartbeats + debug_logs
| fieldsAdd savings_pct = round((toDouble(droppable) / toDouble(total)) *
100, decimals: 1)
```

2. Migration Paths

Path A: Automatic Migration (Recommended)

If you're using OneAgent for log collection, most data automatically flows through OpenPipeline.

Steps:
1. Verify `dt.openpipeline.source` = `oneagent` in your logs
2. Configure custom pipelines for specific processing needs
3. Update queries to use OpenPipeline fields

Path B: API Migration

If using the Log Ingest API:

Steps:
1. Continue using `/api/v2/logs/ingest` endpoint
2. Logs automatically route through OpenPipeline
3. Configure processing rules as needed

Path C: OTLP Migration

For OpenTelemetry-based logging:

Steps:
1. Point OTLP exporters to `/api/v2/otlp/v1/logs`
2. Logs flow through OpenPipeline automatically
3. Configure pipelines for OTLP-specific processing

```python
// Check which data sources are already using OpenPipeline
fetch logs, from: now() - 1h
| filter isNotNull(dt.openpipeline.pipelines)
| summarize {count = count()}, by: {dt.openpipeline.source}
| sort count desc
```

3. Planning Your OpenPipeline Configuration

Pipeline Strategy

| Use Case | Pipeline Configuration |

````
|----------|-----------------------|
| **Default Processing** | Use built-in Default Pipeline |
| **Custom Parsing** | Create pipeline with DPL parse rules |
| **PII Masking** | Add masking stage before processing |
| **Cost Reduction** | Add drop rules for noise |
| **Custom Routing** | Route to specific buckets |

### Bucket Strategy

| Log Type | Bucket | Retention |
|----------|--------|----------|
| Production Errors | `prod_errors` | 90 days |
| Application Logs | `default_logs` | 35 days |
| Debug/Verbose | Drop or 7 days | Minimal |
| Audit/Compliance | `audit_logs` | 365+ days |

```python
// Analyze current bucket usage
fetch logs, from: now() - 24h
| summarize {
    log_count = count(),
    error_count = countIf(loglevel == "ERROR" OR status == "ERROR")
  }, by: {dt.system.bucket}
| fieldsAdd error_rate = round((toDouble(error_count) / toDouble(log_count))
* 100, decimals: 2)
| sort log_count desc
```

## 4. Migration Validation

After configuring OpenPipeline, validate that:

1. ✅ All expected log sources are present
2. ✅ Log counts are consistent
3. ✅ Parsing rules extract expected fields
4. ✅ Masking rules are applied correctly
5. ✅ Routing sends logs to correct buckets

```python
// Validation Query 1: Confirm all sources are flowing
fetch logs, from: now() - 1h
| summarize {
    total = count(),
    has_pipeline = countIf(isNotNull(dt.openpipeline.pipelines)),
    has_source = countIf(isNotNull(dt.openpipeline.source)),
    has_bucket = countIf(isNotNull(dt.system.bucket))
  }
| fieldsAdd pipeline_coverage = round((toDouble(has_pipeline) /
````
````

```
toDouble(total)) * 100, decimals: 1)
```

```python
// Validation Query 2: Check log volume consistency (compare hours)
fetch logs, from: now() - 4h
| makeTimeseries {log_count = count()}, by: {dt.openpipeline.source},
interval: 1h
```

```python
// Validation Query 3: Verify entity context is preserved
fetch logs, from: now() - 1h
| summarize {
    total = count(),
    with_host = countIf(isNotNull(dt.entity.host)),
    with_process = countIf(isNotNull(dt.entity.process_group)),
    with_k8s_cluster = countIf(isNotNull(dt.entity.kubernetes_cluster)),
    with_k8s_namespace = countIf(isNotNull(k8s.namespace.name))
  }
```

5. Common Migration Patterns

bWlkZGxlIj5Db21tb24gTWlncmF0aW9uIFBhdHRlcm5zPC90ZXh0PgogIDx0ZXh0IHg9IjQwMCIge
T0iNDgiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMSIgZmlsbD
0iIzY2NiIgdGV4dC1hbmNob3I9Im1pZGRsZSI+VGhyZWUgZXNzZW50aWFsIE9wZW5QaXBlbGluZSB
wcm9jZXNzb3IgcGF0dGVybnM8L3RleHQ+CgogIDwhLS0gUGF0dGVybiAxOiBQYXJzZSAtLT4KICA8
cmVjdCB4PSIzMCIgeT0iNzAiIHdpZHRoPSIyMzAiIGhlaWdodD0iMjEwIiByeD0iMTAiIGZpbGw9I
iNmZmYiIHN0cm9rZT0iIzNiODJmNiIgc3Ryb2tlLXdpZHRoPSIyIi8+CiAgPHJlY3QgeD0iMzAiIH
k9IjcwIiB3aWR0aD0iMjMwIiBoZWlnaHQ9IjM1IiByeD0iMTAiIGZpbGw9InVybCgjcGF0dGVyblB
hcnNlR3JhZCkiLz4KICA8dGV4dCB4PSIxNDUiIHk9Ijk1IiBmb250LWZhbWlseT0iQXJpYWwsIHNh
bnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTIiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSJ3aGl0ZSIgd
GV4dC1hbmNob3I9Im1pZGRsZSI+UGFyc2UgTG9nIExldmVsczwvdGV4dD4KCiAgPHJlY3QgeD0iNT
AiIHk9IjExNSIgd2lkdGg9IjE5MCIgaGVpZ2h0PSIzNSIgcng9IjYiIGZpbGw9IiNlMmU4ZjAiLz4
KICA8dGV4dCB4PSIxNDUiIHk9IjEzOCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBm
b250LXNpemU9IjEwIiBmaWxsPSIjMzMzIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5SYXcgTG9nIEluc
HV0PC90ZXh0PgoKICA8cGF0aCBkPSJNMTQ1LDE1MCBMMTQ1LDE2NSIgc3Ryb2tlPSIjNjQ3NDhiIi
BzdHJva2Utd2lkdGg9IjIiIGZpbGw9Im5vbmUiIG1hcmtlci1lbmQ9InVybCgjcGF0dGVybkFycm9
3KSIvPgoKICA8cmVjdCB4PSI1MCIgeT0iMTc1IiB3aWR0aD0iMTkwIiBoZWlnaHQ9IjUwIiByeD0i
NiIgZmlsbD0idXJsKCNwYXR0ZXJuUGFyc2VHcmFkKSIgZmlsdGVyPSJ1cmwoI3BhdHRlcm5TaGFkb
3cpIi8+CiAgPHRleHQgeD0iMTQ1IiB5PSIxOTUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZX
JpZiIgZm9udC1zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9IndoaXRlIiB0ZXh0LWF
uY2hvcj0ibWlkZGxlIj5EUEwgUHJvY2Vzc29yPC90ZXh0PgogIDx0ZXh0IHg9IjE0NSIgeT0iMjE1
IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyN
TUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPldPUkQ6bG9nbGV2ZWwgJyAnIExxEPC90ZX
h0PgoKICA8cGF0aCBkPSJNMTQ1LDIyNSBMMTQ1LDI0MCIgc3Ryb2tlPSIjNjQ3NDhiIiBzdHJva2U
td2lkdGg9IjIiIGZpbGw9Im5vbmUiIG1hcmtlci1lbmQ9InVybCgjcGF0dGVybkFycm93KSIvPgoK
ICA8cmVjdCB4PSI1MCIgeT0iMjUwIiB3aWR0aD0iMTkwIiBoZWlnaHQ9IjI1IiByeD0iNiIgZmlsb
D0iI2QxZmFlNSIvPgogIDx0ZXh0IHg9IjE0NSIgeT0iMjY3IiBmb250LWZhbWlseT0iQXJpYWwsIH
NhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiMwNDc4NTciIHRleHQtYW5jaG9yPSJtaWR
kbGUiPmxvZ2xldmVsIGZpZWxkIGV4dHJhY3RlZDwvdGV4dD4KCiAgPCEtLSBQYXR0ZXJuIDI6IERy
b3AgLS0+CiAgPHJlY3QgeD0iMjg1IiB5PSI3MCIgd2lkdGg9IjIzMCIgaGVpZ2h0PSIyMTAiIHJ4P
SIxMCIgZmlsbD0iI2ZmZiIgc3Ryb2tlPSIjZWY0NDQ0IiBzdHJva2Utd2lkdGg9IjIiLz4KICA8cm
VjdCB4PSIyODUiIHk9IjcwIiB3aWR0aD0iMjMwIiBoZWlnaHQ9IjM1IiByeD0iMTAiIGZpbGw9InV
ybCgjcGF0dGVybkRyb3BHcmFkKSIvPgogIDx0ZXh0IHg9IjQwMCIgeT0iOTUiIGZvbnQtZmFtaWx5
PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMiIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpb
Gw9IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5Ecm9wIERlYnVnIExvZ3M8L3RleHQ+CgogID
xyZWN0IHg9IjMwNSIgeT0iMTE1IiB3aWR0aD0iODUiIGhlaWdodD0iMzUiIHJ4PSI2IiBmaWxsPSI
jZTJlOGYwIi8+CiAgPHRleHQgeD0iMzQ3IiB5PSIxMzgiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fu
cy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzMzMyIgdGV4dC1hbmNob3I9Im1pZGRsZSI+Q
WxsIExvZ3M8L3RleHQ+CgogIDxyZWN0IHg9IjQxMCIgeT0iMTE1IiB3aWR0aD0iODUiIGhlaWdodD
0iMzUiIHJ4PSI2IiBmaWxsPSIjZDFmYWU1Ii8+CiAgPHRleHQgeD0iNDUyIiB5PSIxMzgiIGZvbnQ
tZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzA0Nzg1NyIg
dGV4dC1hbmNob3I9Im1pZGRsZSI+S2VlcDwvdGV4dD4KCiAgPHBhdGggZD0iTTM0NywxNTAgTDQwM
CwxOTAiIHN0cm9rZT0iI2VmNDQ0NCIgc3Ryb2tlLXdpZHRoPSIyIiBmaWxsPSJub25lIiBtYXJrZX
ItZW5kPSJ1cmwoI3BhdHRlcm5BcnJvdykiLz4KICA8cGF0aCBkPSJNNDUyLDE1MCBMNDAwLDE5MCI
gc3Ryb2tlPSIjMTBiOTgxIiBzdHJva2Utd2lkdGg9IjIiIGZpbGw9Im5vbmUiIG1hcmtlci1lbmQ9
InVybCgjcGF0dGVybkFycm93KSIvPgoKICA8cmVjdCB4PSIzMDUiIHk9IjE3NSIgd2lkdGg9IjE5M
CIgaGVpZ2h0PSI1MCIgcng9IjYiIGZpbGw9InVybCgjcGF0dGVybkRyb3BHcmFkKSIgZmlsdGVyPS
J1cmwoI3BhdHRlcm5TaGFkb3cpIi8+CiAgPHRleHQgeD0iNDAwIiB5PSIxOTUiIGZvbnQtZmFtaWx
5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZp

bGw9IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5Ecm9wIFByb2Nlc3NvcjwvdGV4dD4KICA8d
GV4dCB4PSI0MDAiIHk9IjIxNSIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMC
IgZmlsbD0icmdiYSgyNTUsMjU1LDI1NSwwLjkpIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5sb2dsZXZ
lbCA9PSAiREVCVUciPC90ZXh0PgoKICA8cGF0aCBkPSJNNDAwLDIyNSBMNDAwLDI0MCIgc3Ryb2tl
PSIjNjQ3NDhiIiBzdHJva2Utd2lkdGg9IjIiIGZpbGw9Im5vbmUiG1hcmtlci1lbmQ9InVybCgjc
GF0dGVybkFycm93KSIvPgoKICA8cmVjdCB4PSIzMDUiHk9IjI1MCIgd2lkdGg9IjE5MCIgaGVpZ2
h0PSIyNSIgcng9IjYiIGZpbGw9IiNmZWYyZjIiLz4KICA8dGV4dCB4PSI9MDAiHk9IjI2NyIgZm9
udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjZGMyNjI2
IiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj4zNSUgbG9ncyBkcm9wcGVkIOKGkiBzYXZpbmdzITwvdGV4d
D4KCiAgPCEtLSBQYXR0ZXJuIDM6IFJvdXRlIC0tPgogIDx0ZXh0IHg9IjEwIiBoeT0iNzAiIHdpZH
RoPSIyMzUiIGZpbGw9IiNmZmYiIHN0cm9rZT0iIzEwYjk4MSI gc3Ryb2tlLXdpZHRoPSIxIiB5PSIxM
zgiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIz
MzMyIgdGV4dC1hbmNob3I9Im1pZGRsZSI+RXJyb3IgTG9nczwvdGV4dD4KICAgPHBhdGggZD0iTTY
1NSwxNTAgTDY1NSwxNjUiHN0cm9rZT0iIzY0NzQ4YiIgc3Ryb2tlLXdpZHRoPSIxIiB5PSIxMC
b25lIiBtYXJrZXItZW5kPSJ1cmwoI3BhdHRlcm5BcnJvdykiLz4KICAgPHJlY3QgeD0iNTYwIiHk
SIxNzUiHdpZHRoPSIxOTAiIGhlaWdodD0iNTAiIHJ4PSI2IiBmaWxsPSJ1cmwoI3BhdHRlcm5Sb3
V0ZUdyYWQpIiBmaWx0ZXI9InVybCgjcGF0dGVyblNoYWRvdykiLz4KICA8dGV4dCB4PSI2NTUiIHk
9IjE5NSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjExIiBmb250
LXdlaWdodD0iYm9sZCIgZmlsbD0id2hpdGUiIHRleHQtYW5jaG9yPSJtaWRkbGUiPkJ1Y2tldCBSb
3V0ZXI8L3RleHQ+CiAgPHRleHQgeD0iNTYwIiB5PSIyMTUiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2
UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I
9Im1pZGRsZSI+4oaSIGVycm9yX2xvZ3MgYnVja2V0PC90ZXh0PgoKICA8cGF0aCBkPSJNNjAwLDIy
NSBMNTYwLDI1MCIgc3Ryb2tlPSIjNjQ3NDhiIiBzdHJva2Utd2lkdGg9IjIiIGZpbGw9Im5vbmUiI
G1hcmtlci1lbmQ9InVybCgjcGF0dGVybkFycm93KSIvPgogIDxwogIDxwYXRoIGQ9Ik03MTAsMjI1IEw3NT
AsMjUwIiBzdHJva2U9IiM2NDc0OGIiHN0cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgbWFya2V
yLWVuZD0idXJsKCNwYXR0ZXJuQXJyb3cpIi8+CgogIDxwYXRoIGQ9Ik03MTAsMjI1IEw3NTAsMj
UwIiBzdHJva2U9IiM2NDc0OGIiHN0cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgbWFya2VyLW
VuZD0idXJsKCNwYXR0ZXJuQXJyb3cpIi8+CgogIDxwYXRoIGQ9Ik03MTAsMjI1IEw3NTAsMjUwIiB
zdHJva2U9IiM2NDc0OGIiHN0cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgbWFya2VyLWVuZD0i
dXJsKCNwYXR0ZXJuQXJyb3cpIi8+CgogIDxwYXRoIGQ9Ik03MTAsMjI1IEw3NTAsMjUwIiBzdHJva
2U9IiM2NDc0OGIiHN0cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgbWFya2VyLWVuZD0idXJs
KCNwYXR0ZXJuQXJyb3cpIi8+CgogIDxwYXRoIGQ9Ik03MTAsMjI1IEw3NTAsMjUwIiBzdHJva2U9
IiM2NDc0OGIiHN0cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgbWFya2VyLWVuZD0idXJsKCNw
YXR0ZXJuQXJyb3cpIi8+CgogIDxwYXRoIGQ9Ik03MTAsMjI1IEw3NTAsMjUwIiBzdHJva2U9IiM2
NDc0OGIiHN0cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgbWFya2VyLWVuZD0idXJsKCNwYXR0
ZXJuQXJyb3cpIi8+CgogIDxwYXRoIGQ9Ik03MTAsMjI1IEw3NTAsMjUwIiBzdHJva2U9IiM2NDc0
OGIiHN0cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgbWFya2VyLWVuZD0idXJsKCNwYXR0ZXJu
QXJyb3cpIi8+CgogIDxwYXRoIGQ9Ik03MTAsMjI1IEw3NTAsMjUwIiBzdHJva2U9IiM2NDc0OGIi
HN0cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgbWFya2VyLWVuZD0idXJsKCNwYXR0ZXJuQXJy
b3cpIi8+CgogIDxwYXRoIGQ9Ik03MTAsMjI1IEw3NTAsMjUwIiBzdHJva2U9IiM2NDc0OGIiHN0
cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgbWFya2VyLWVuZD0idXJsKCNwYXR0ZXJuQXJyb3c
pIi8+CgogIDxwYXRoIGQ9Ik03MTAsMjI1IEw3NTAsMjUwIiBzdHJva2U9IiM2NDc0OGIiHN0cm9r
ZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgbWFya2VyLWVuZD0idXJsKCNwYXR0ZXJuQXJyb3cpIi8
+CiAgPC9zdmc+)

Pattern 1: Parse Log Levels from Content

When logs have `loglevel = NONE`, configure a DQL processor to extract the level from content patterns like `[INFO]` or `[ERROR]`.

Processor Configuration:

````
- **Type:** DQL
- **Matcher:** `loglevel == "NONE"`
- **Statement:** `parse content, "'[' LD:parsed_level ']'" | fieldsAdd
loglevel = parsed_level`

### Pattern 2: Drop Debug Logs

Reduce storage costs by dropping DEBUG-level logs before storage.

**Processor Configuration:**
- **Type:** Drop
- **Matcher:** `loglevel == "DEBUG"`

### Pattern 3: Route Errors to Dedicated Bucket

Send error logs to a dedicated bucket with longer retention for compliance.

**Bucket Routing Configuration:**
- **Matcher:** `loglevel == "ERROR"`
- **Bucket:** `error_logs`
- **Retention:** 90 days

```python
// Simulate parsing log levels from content
fetch logs, from: now() - 1h
| filter loglevel == "NONE" OR status == "NONE"
| parse content, "'[' LD:parsed_level ']'"
| filter isNotNull(parsed_level)
| fields timestamp, content, parsed_level
| limit 10
```

```python
// Identify logs that would be dropped (DEBUG + health checks)
fetch logs, from: now() - 24h
| filter loglevel == "DEBUG"
     OR matchesPhrase(content, "healthcheck")
     OR matchesPhrase(content, "health check")
| summarize {would_drop = count()}, by: {dt.openpipeline.source}
| sort would_drop desc
```

## 6. Migration Checklist

### Pre-Migration
- [ ] Document current log sources and volumes
- [ ] Identify logs requiring custom parsing
- [ ] List sensitive data patterns to mask
````

```
- [ ] Define drop rules for noise reduction
- [ ] Plan bucket strategy and retention

### During Migration
- [ ] Configure OpenPipeline settings
- [ ] Create custom pipelines as needed
- [ ] Set up masking rules
- [ ] Configure bucket routing
- [ ] Test with sample data

### Post-Migration
- [ ] Validate all log sources flowing
- [ ] Verify volume consistency
- [ ] Confirm parsing rules working
- [ ] Test masking is applied
- [ ] Update dashboards and alerts
- [ ] Update saved queries to use new fields

---

## 📝 Summary

In this notebook, you learned:

✅ **Assessment queries** to understand your current log environment
✅ **Migration paths** for OneAgent, API, and OTLP sources
✅ **Pipeline planning** strategies for processing and routing
✅ **Validation queries** to confirm successful migration
✅ **Common patterns** for parsing, dropping, and routing

---

## ➡️ Next Steps

Continue to **OPLOGS-03: Querying & Parsing** to learn DQL syntax and DPL parsing patterns.

---

## 📚 References

- [OpenPipeline Documentation](https://docs.dynatrace.com/docs/discover-dynatrace/platform/openpipeline)
- [Log Ingest API](https://docs.dynatrace.com/docs/dynatrace-api/environment-api/log-monitoring-v2/post-ingest-logs)
- [OTLP Log Ingestion](https://docs.dynatrace.com/docs/extend-dynatrace/opentelemetry/getting-started/logs)
```