# 🗄️ Buckets & Data Governance

> **Series:** OPLOGS | **Notebook:** 4 of 8 | **Created:** December 2025

## Strategic Storage Management for OpenPipeline Logs

This notebook covers Grail bucket architecture, retention policies, routing configuration, access control, and cost optimization strategies.

## Table of Contents

## Prerequisites

- ✅ Access to a Dynatrace environment with log data
- ✅ Understanding of OpenPipeline basics (OPLOGS-01 to OPLOGS-03)
- ✅ Admin access for bucket configuration (optional)

```python
// Discover all buckets with log data
fetch logs, from: now() - 24h
| summarize {log_count = count()}, by: {dt.system.bucket}
| sort log_count desc
```

```python
// Bucket volume analysis over time
fetch logs, from: now() - 7d
| makeTimeseries {count = count()}, by: {dt.system.bucket}, interval: 1d
| limit 100
```

```python
// Bucket composition by log level
fetch logs, from: now() - 24h
```

~~~~
| summarize {count = count()}, by: {dt.system.bucket, loglevel}
| sort dt.system.bucket asc, count desc
```

## 2. Strategic Bucket Design

Design buckets based on **access patterns**, **retention requirements**, and
**cost optimization**.

![Bucket Governance Strategy]
~~~~

(data:image/svg+xml;base64,PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdm
ciIHZpZXdCb3g9IjAgMCA4MDAgMzQwIj4KICA8ZGVmcz4KICAgIDxsaW5lYXJHcmFkaWVudCBpZD0
iZXJyb3JCdWNrZXQiIHgxPSIwJSIgeTE9IjAlIiB4Mj0iMTAwJSIgeTI9IjEwMCUiPgogICAgICA8
c3RvcCBvZmZzZXQ9IjAlIiBzdHlsZT0ic3RvcC1jb2xvcjojZWY0NDQ0O3N0b3Atb3BhY2l0eToxI
iAvPgogICAgICA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiNkYzI2MjY7c3
RvcC1vcGFjaXR5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50PgogICAgPGxpbmVhckdyYWRpZW5
0IGlkPSJ3YXJuQnVja2V0IiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIxMDAlIj4KICAg
ICAgPHN0b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6I2Y1OWUwYjtzdG9wLW9wYWNpd
Hk6MSIgLz4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcjojZDk3Nz
A2O3N0b3Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxsaW5lYXJHcmF
kaWVudCBpZD0iaW5mb0J1Y2tldCIgeDE9IjAlIiB5MT0iMCUiIHgyPSIxMDAlIiB5Mj0iMTAwJSI+
CiAgICAgIDxzdG9wIG9mZnNldD0iMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiMzYjgyZjY7c3RvcC1vc
GFjaXR5OjEiIC8+CiAgICAgIDxzdG9wIG9mZnNldD0iMTAwJSIgc3R5bGU9InN0b3AtY29sb3I6Iz
I1NjNlYjtzdG9wLW9wYWNpdHk6MSIgLz4KICAgIDwvbGluZWFyR3JhZGllbnQ+CiAgICA8bGluZWF
yR3JhZGllbnQgaWQ9ImF1ZGl0QnVja2V0IiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIx
MDAlIj4KICAgICAgPHN0b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6IzhiNWNmNjtzd
G9wLW9wYWNpdHk6MSIgLz4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2
xvcjojN2MzYWVkO3N0b3Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDx
saW5lYXJHcmFkaWVudCBpZD0iZGVidWdCdWNrZXQiIHgxPSIwJSIgeTE9IjAlIiB4Mj0iMTAwJSIg
eTI9IjEwMCUiPgogICAgICA8c3RvcCBvZmZzZXQ9IjAlIiBzdHlsZT0ic3RvcC1jb2xvcjojMTBiiO
TgxO3N0b3Atb3BhY2l0eToxIiAvPgogICAgICA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG
9wLWNvbG9yOiMwNTk2Njk7c3RvcC1vcGFjaXR5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50Pgo
gICAgPGZpbHRlciBpZD0iYnVja2V0U2hhZG93Ij4KICAgICAgPGZlRHJvcFNoYWRvdyBkeD0iMSIg
ZHk9IjEiIHN0ZERldmlhdGlvbj0iMiIgZmxvb2Qtb3BhY2l0eT0iMC4xNSIvPgogICAgPC9maWx0Z
XI+CiAgICA8bWFya2VyIGlkPSJidWNrZXRBcnJvdyIgbWFya2VyV2lkdGg9IjEwIiBtYXJrZXJIZW
lnaHQ9IjciIHJlZlg9IjkiIHJlZlk9IjMuNSIgb3JpZW50PSJhdXRvIj4KICAgICAgPHBvbHlnb24
gcG9pbnRzPSIwIDAsIDEwIDMuNSwgMCA3IiBmaWxsPSIjNjQ3NDhiIi8+CiAgICA8L21hcmtlcj4K
ICA8L2RlZnM+CgogIDwhLS0gQmFja2dyb3VuZCAtLT4KICA8cmVjdCB3aWR0aD0iODAwIiBoZWlna
HQ9IjM0MCIgZmlsbD0iI2Y4ZjlmYSIgcng9IjEwIi8+CgogIDwhLS0gVGl0bGUgLS0+CiAgPHRleH
QgeD0iNDAwIiB5PSIyOCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU
9IjE4IiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0iIzMzMyIgdGV4dC1hbmNob3I9Im1pZGRsZSI+
QnVja2V0IEdvdmVybmFuY2UgU3RyYXRlZ3k8L3RleHQ+CiAgPHRleHQgeD0iNDAwIiB5PSI0OCIgZ
m9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjExIiBmaWxsPSIjNjY2Ii
B0ZXh0LWFuY2hvcj0ibWlkZGxlIj5Sb3V0ZSBsb2dzIHRvIHB1cnBvc2UtYnVpbHQgYnVja2V0cyB
3aXRoIHRpZXJlZCByZXRlbnRpb248L3RleHQ+CgogIDwhLS0gTG9nIFNvdXJjZXMgLS0+CiAgPHJl
Y3QgeD0iMzAiIHk9IjcwIiB3aWR0aD0iMTAwIiBoZWlnaHQ9IjIwMCIgcng9IjEwIiBmaWxsPSIjZ
mZmIiBzdHJva2U9IiNlMmU4ZjAiIHN0cm9rZS13aWR0aD0iMiIvPgogIDx0ZXh0IHg9IjgwIiB5PS
I5NSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjExIiBmb250LXd

```
laWdodD0iYm9sZCIgZmlsbD0iIzMzMyIgdGV4dC1hbmNob3I9Im1pZGRsZSI+QWxsIExvZ3M8L3Rl
eHQ+CiAgPHRleHQgeD0iODAiIHk9IjExNSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmI
iBmb250LXNpemU9IjEwIiBmaWxsPSIjNjQ3NDhiIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj4xMDAlPC
90ZXh0PgogIDx0ZXh0IHg9IjgwIiB5PSIxMzUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJ
pZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzY0NzQ4YiIgdGV4dC1hbmNob3I9Im1pZGRsZSI+dm9s
dW1lPC90ZXh0PgoKICA8cmVjdCB4PSI1MCIgeT0iMTU1IiB3aWR0aD0iNjAiIGhlaWdodD0iMTUiI
HJ4PSIzIiBmaWxsPSIjZWY0NDQ0Ii8+CiAgPHRleHQgeD0iODAiIHk9IjE2NiIgZm9udC1mYW1pbH
k9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmN
ob3I9Im1pZGRsZSI+NSU8L3RleHQ+CgogIDxyZWN0IHg9IjUwIiB5PSIxNzUiIHdpZHRoPSI2MCIg
aGVpZ2h0PSIxNSIgcng9IjMiIGZpbGw9IiNmNTllMGIiLz4KICA8dGV4dCB4PSI4MCIgeT0iMTg2I
iBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IndoaX
RlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj4xMCU8L3RleHQ+CgogIDxyZWN0IHg9IjUwIiB5PSIxOTU
iIHdpZHRoPSI2MCIgaGVpZ2h0PSIxNSIgcng9IjMiIGZpbGw9IiMzYjgyZjYiLz4KICA8dGV4dCB4
PSI4MCIgeT0iMjA2IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iM
TAiIGZpbGw9IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj40MCU8L3RleHQ+CgogIDxyZWN0IH
g9IjUwIiB5PSIyMTUiIHdpZHRoPSI2MCIgaGVpZ2h0PSIxNSIgcng9IjMiIGZpbGw9IiM4YjVjZjY
iLz4KICA8dGV4dCB4PSI4MCIgeT0iMjI2IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYi
IGZvbnQtc2l6ZT0iMTAiIGZpbGw9IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj4xMCU8L3Rle
HQ+CgogIDxyZWN0IHg9IjUwIiB5PSIyMzUiIHdpZHRoPSI2MCIgaGVpZ2h0PSIxNSIgcng9IjMiIG
ZpbGw9IiMxMGI5ODEiLz4KICA8dGV4dCB4PSI4MCIgeT0iMjQ2IiBmb250LWZhbWlseT0iQXJpYWw
sIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlk
ZGxlIj4zNSU8L3RleHQ+CgogIDwhLS0gUm91dGluZyBBcnJvd3MgLS0+CiAgPHBhdGggZD0iTTEzM
CwxNjIgTDE3MCwxMDAiIHN0cm9rZT0iI2VmNDQ0NCIgc3Ryb2tlLXdpZHRoPSIyIiBmaWxsPSJub2
5lIiBtYXJrZXItZW5kPSJ1cmwoI2J1Y2tldEFycm93KSIvPgogIDxwYXRoIGQ9Ik0xMzAsMTgyIEw
xNzAsMTQ1IiBzdHJva2U9IiNmNTllMGIiIHN0cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgbWFy
a2VyLWVuZD0idXJsKCNidWNrZXRBcnJvdykiLz4KICA8cGF0aCBkPSJNMTMwLDIwMiBMMTcwLDE5M
CIgc3Ryb2tlPSIjM2I4MmY2IiBzdHJva2Utd2lkdGg9IjIiIGZpbGw9Im5vbmUiIG1hcmtlci1lbm
Q9InVybCgjYnVja2V0QXJyb3cpIi8+CiAgPHBhdGggZD0iTTEzMCwyMjIgTDE3MCwyMzUiIHN0cm9
rZT0iIzhiNWNmNiIgc3Ryb2tlLXdpZHRoPSIyIiBmaWxsPSJub25lIiBtYXJrZXItZW5kPSJ1cmwo
I2J1Y2tldEFycm93KSIvPgogIDxwYXRoIGQ9Ik0xMzAsMjQyIEwxNzAsMjgwIiBzdHJva2U9IiMxM
GI5ODEiIHN0cm9rZS13aWR0aD0iMiIgZmlsbD0ibm9uZSIgbWFya2VyLWVuZD0idXJsKCNidWNrZX
RBcnJvdykiLz4KCiAgPCEtLSBCdWNrZXRzIC0tPgogIDxyZWN0IHg9IjE4MCIgeT0iNzUiIHdpZHR
oPSIxODAiIGhlaWdodD0iNDUiIHJ4PSI4IiBmaWxsPSJ1cmwoI2Vycm9yQnVja2V0KSIgZmlsdGVy
PSJ1cmwoI2J1Y2tldFNoYWRvdykiLz4KICA8dGV4dCB4PSIyNzAiIHk9Ijk1IiBmb250LWZhbWlse
T0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaW
xsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+ZXJyb3JfbG9nczwvdGV4dD4KICA8dGV4dCB
4PSIyNzAiIHk9IjExMiIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9
IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPjkwL
WRheSByZXRlbnRpb248L3RleHQ+CgogIDxyZWN0IHg9IjE4MCIgeT0iMTMwIiB3aWR0aD0iMTgwIi
BoZWlnaHQ9IjQ1IiByeD0iOCIgZmlsbD0idXJsKCN3YXJuQnVja2V0KSIgZmlsdGVyPSJ1cmwoI2J
1Y2tldFNoYWRvdykiLz4KICA8dGV4dCB4PSIyNzAiIHk9IjE1MCIgZm9udC1mYW1pbHk9IkFyaWFs
LCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0id2hpd
GUiIHRleHQtYW5jaG9yPSJtaWRkbGUiPndhcm5pbmdfbG9nczwvdGV4dD4KICA8dGV4dCB4PSIyNz
AiIHk9IjE2NyIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiB
maWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPjM1LWRheSBy
ZXRlbnRpb248L3RleHQ+CgogIDxyZWN0IHg9IjE4MCIgeT0iMTg1IiB3aWR0aD0iMTgwIiBoZWlna
HQ9IjQ1IiByeD0iOCIgZmlsbD0idXJsKCNpbmZvQnVja2V0KSIgZmlsdGVyPSJ1cmwoI2J1Y2tldF
NoYWRvdykiLz4KICA8dGV4dCB4PSIyNzAiIHk9IjIwNSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5
```

```
zLXNlcmlmIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0id2hpdGUiIHRl
eHQtYW5jaG9yPSJtaWRkbGUiPmRlZmF1bHRfbG9nczwvdGV4dD4KICA8dGV4dCB4PSIyNzAiIHk9I
jIyMiIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPS
JyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPjM1LWRheSByZXRlbnR
pb248L3RleHQ+CgogIDxyZWN0IHg9IjE4MCIgeT0iMjQwIiB3aWR0aD0iMTgwIiBoZWlnaHQ9IjQ1
IiByeD0iOCIgZmlsbD0idXJsKCNhdWRpdEJ1Y2tldCkiIGZpbHRlcj0idXJsKCNidWNrZXRTaGFkb
3cpIi8+CiAgPHRleHQgeD0iMjcwIiB5PSIyNjAiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZX
JpZiIgZm9udC1zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9IndoaXRlIiB0ZXh0LWF
uY2hvcj0ibWlkZGxlIj5hdWRpdF9sb2dzPC90ZXh0PgogIDx0ZXh0IHg9IjI3MCIgeT0iMjc3IiBm
b250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoM
jU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+MS15ZWFyIHJldGVudGlvbjwvdG
V4dD4KCiAgPHJlY3QgeD0iMTgwIiB5PSIyOTUiIHdpZHRoPSIxODAiIGhlaWdodD0iNDUiIHJ4PSI
4IiBmaWxsPSJ1cmwoI2RlYnVnQnVja2V0KSIgZmlsdGVyPSJ1cmwoI2J1Y2tldFNoYWRvdykiLz4K
ICA8dGV4dCB4PSIyNzAiIHk9IjMxNSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb
250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0id2hpdGUiIHRleHQtYW5jaG9yPS
JtaWRkbGUiPmRlYnVnX2xvZ3M8L3RleHQ+CiAgPHRleHQgeD0iMjcwIiB5PSIzMzIiIGZvbnQtZmF
taWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0icmdiYSgyNTUsMjU1
LDI1NSwwLjkpIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj43LWRheSByZXRlbnRpb24gKG9yIERFST1ApP
C90ZXh0PgoKICA8IS0tIEJlbmVmaXRzIFNlY3Rpb24gLS0+CiAgPHJlY3QgeD0iNDAwIiB5PSI3MC
Igd2lkdGg9IjM3MCIgaGVpZ2h0PSIyNTAiIHJ4PSIxMCIgZmlsbD0iI2ZmZiIgc3Ryb2tlPSIjZTJ
lOGYwIiBzdHJva2Utd2lkdGg9IjIiLz4KICA8dGV4dCB4PSI1ODUiIHk9Ijk1IiBmb250LWZhbWls
eT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTIiIGZvbnQtd2VpZ2h0PSJib2xkIiBma
WxsPSIjMzMzIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5Hb3Zlcm5hbmNlIEJlbmVmaXRzPC90ZXh0Pg
oKICA8cmVjdCB4PSI0MjAiIHk9IjExMCIgd2lkdGg9IjE2NSIgaGVpZ2h0PSI5MCIgcng9IjYiIGZ
pbGw9IiNmZWYyZjIiLz4KICA8dGV4dCB4PSI1MDIiIHk9IjEzNSIgZm9udC1mYW1pbHk9IkFyaWFs
LCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0iI2RjM
jYyNiIgdGV4dC1hbmNob3I9Im1pZGRsZSI+Q29zdCBDb250cm9sPC90ZXh0PgogIDx0ZXh0IHg9Ij
UwMiIgeT0iMTU1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTA
iIGZpbGw9IiM3ZjFkMWQiIHRleHQtYW5jaG9yPSJtaWRkbGUiPkRST1AgZGVidWcgPSAzNSU8L3Rl
eHQ+CiAgPHRleHQgeD0iNTAyIiB5PSIxNzAiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZ
iIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzdmMWQxZCIgdGV4dC1hbmNob3I9Im1pZGRsZSI+c2F2aW
5nczwvdGV4dD4KICA8dGV4dCB4PSI1MDIiIHk9IjE5MCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5
zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjN2YxZDFkIiB0ZXh0LWFuY2hvcj0ibWlkZGxl
Ij5TaG9ydCByZXRlbnRpb24gPSBsZXNzIEREVTwvdGV4dD4KCiAgPHJlY3QgeD0iNTk1IiB5PSIxM
TAiIHdpZHRoPSIxNjUiIGhlaWdodD0iOTAiIHJ4PSI2IiBmaWxsPSIjZDFmYWU1Ii8+CiAgPHRleH
QgeD0iNjc3IiB5PSIxMzUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXp
lPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9IiMwNDc4NTciIHRleHQtYW5jaG9yPSJtaWRk
bGUiPkNvbXBsaWFuY2U8L3RleHQ+CiAgPHRleHQgeD0iNjc3IiB5PSIxNTUiIGZvbnQtZmFtaWx5P
SJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzA2NGUzYiIgdGV4dC1hbm
Nob3I9Im1pZGRsZSI+QXVkaXQgbG9nczogMSB5ZWFyPC90ZXh0PgogIDx0ZXh0IHg9IjY3NyIgeT0
iMTcwIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9
IiMwNjRlM2IiIHRleHQtYW5jaG9yPSJtaWRkbGUiPlBDSS9ISVBBQSByZWFkeTwvdGV4dD4KICA8d
GV4dCB4PSI2NzciIHk9IjE5MCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LX
NpemU9IjEwIiBmaWxsPSIjMDY0ZTNiIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5TZXBhcmF0aW9uIG9
mIGNvbmNlcm5zPC90ZXh0PgoKICA8cmVjdCB4PSI0MjAiIHk9IjIxMCIgd2lkdGg9IjE2NSIgaGVp
Z2h0PSI5MCIgcng9IjYiIGZpbGw9IiNkYmVhZmUiLz4KICA8dGV4dCB4PSI1MDIiIHk9IjIzNSIgZ
m9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD
0iYm9sZCIgZmlsbD0iIzFlNDBhZiIgdGV4dC1hbmNob3I9Im1pZGRsZSI+QWNjZXNzIENvbnRyb2w
```

~~~~
8L3RleHQ+CiAgPHRleHQgeD0iNTAyIiB5PSIyNTUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1z
ZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzFlM2E4YSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+V
GVhbS1iYXNlZCBidWNrZXRzPC90ZXh0PgogIDx0ZXh0IHg9IjUwMiIgeT0iMjcwIiBmb250LWZhbW
lseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiMxZTNhOGEiIHRleHQ
tYW5jaG9yPSJtaWRkbGUiPklBTSBwb2xpY2llcyBwZXIgYnVja2V0PC90ZXh0PgogIDx0ZXh0IHg9
IjUwMiIgeT0iMjkwIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iM
TAiIGZpbGw9IiMxZTNhOGEiIHRleHQtYW5jaG9yPSJtaWRkbGUiPk5lZWQtdG8ta25vdyBhY2Nlc3
M8L3RleHQ+CgogIDxyZWN0IHg9IjU5NSIgeT0iMjEwIiB3aWR0aD0iMTY1IiBoZWlnaHQ9IjkwIiB
yeD0iNiIgZmlsbD0iI2ZlZjNjNyIvPgogIDx0ZXh0IHg9IjY3NyIgeT0iMjM1IiBmb250LWZhbWls
eT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZvbnQtd2VpZ2h0PSJib2xkIiBma
WxsPSIjOTI0MDBlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5QZXJmb3JtYW5jZTwvdGV4dD4KICA8dG
V4dCB4PSI2NzciIHk9IjI1NSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXN
pemU9IjEwIiBmaWxsPSIjNzgzNTBmIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5GYXN0ZXIgcXVlcmll
cyBvbjwvdGV4dD4KICA8dGV4dCB4PSI2NzciIHk9IjI3MCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzY
W5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNzgzNTBmIiB0ZXh0LWFuY2hvcj0ibWlkZG
xlIj5zbWFsbGVyIGJ1Y2tldHM8L3RleHQ+CiAgPHRleHQgeD0iNjc3IiB5PSIyOTAiIGZvbnQtZmF
taWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzc4MzUwZiIgdGV4
dC1hbmNob3I9Im1pZGRsZSI+UmVkdWNlZCBzY2FuIGNvc3Q8L3RleHQ+Cjwvc3ZnPgo=)
~~~~

~~~~
### Recommended Bucket Strategy

| Bucket Name | Purpose | Retention | Access | Data Types |
|-------------|---------|-----------|--------|------------|
| `default_logs` | General purpose | 35 days | All teams | Standard logs |
| `debug_logs` | Development troubleshooting | 7 days | Dev teams | DEBUG, TRACE |
| `error_logs` | Incident investigation | 90 days | SRE/Ops | ERROR, FATAL |
| `audit_logs` | Compliance | 365 days | Audit team | Auth, access |
| `security_logs` | Security operations | 180 days | SecOps | Security events |
| `archive_logs` | Long-term storage | 730 days | Compliance | Critical events |

### Bucket Naming Convention

```
_logs
__logs
__logs
```

Examples:
- `platform_debug_logs`
- `prod_security_logs`
- `payment_audit_logs`
~~~~

```python
// Analyze current log distribution for bucket planning
fetch logs, from: now() - 24h
| summarize {
    total = count(),
    debug = countIf(loglevel == "DEBUG" OR loglevel == "TRACE"),
    info = countIf(loglevel == "INFO" OR loglevel == "NOTICE"),
    warn = countIf(loglevel == "WARN" OR loglevel == "WARNING"),
    error = countIf(loglevel == "ERROR" OR loglevel == "FATAL")
  }
| fieldsAdd debug_pct = round((debug * 100.0) / total, decimals: 1)
| fieldsAdd info_pct = round((info * 100.0) / total, decimals: 1)
| fieldsAdd warn_pct = round((warn * 100.0) / total, decimals: 1)
| fieldsAdd error_pct = round((error * 100.0) / total, decimals: 1)
```

```python
// Identify audit/security logs for specialized bucket
fetch logs, from: now() - 24h
| filter contains(content, "auth")
        OR contains(content, "login")
        OR contains(content, "logout")
        OR contains(content, "permission")
        OR contains(content, "access")
| summarize {count = count()}, by: {k8s.namespace.name, loglevel}
| sort count desc
| limit 20
```

```python
// Volume by source - helps determine routing needs
fetch logs, from: now() - 24h
| summarize {count = count()}, by: {dt.openpipeline.source, dt.system.bucket}
| sort count desc
```

## 3. Retention Policies

### Retention Strategy Matrix

Data Type	Min Retention	Recommended	Max	Rationale
DEBUG/TRACE	3 days	7 days	14 days	High volume, temporary
INFO	7 days	35 days	90 days	Standard operations
WARN	14 days	60 days	180 days	Trend analysis
ERROR	30 days	90 days	365 days	Incident investigation
Audit	365 days	365 days	730 days	Compliance
Security	90 days	180 days	365 days	SIEM/forensics

~~~
### Retention Cost Impact

```
Storage Cost = Log Volume × Retention Days × Cost per GB-day

Example:
- 100 GB/day DEBUG logs × 35 days = 3,500 GB stored
- 100 GB/day DEBUG logs × 7 days  = 700 GB stored
- Savings: 80% reduction in DEBUG storage
```

```python
// Estimate storage by log level
fetch logs, from: now() - 24h
| fieldsAdd content_bytes = stringLength(content)
| summarize {
    log_count = count(),
    total_mb = round(sum(content_bytes) / 1048576.0, decimals: 2),
    avg_bytes = round(avg(content_bytes), decimals: 0)
  }, by: {loglevel}
| sort total_mb desc
```

```python
// Calculate retention cost comparison (DEBUG logs)
fetch logs, from: now() - 24h
| filter loglevel == "DEBUG" OR loglevel == "TRACE"
| fieldsAdd content_bytes = stringLength(content)
| summarize {daily_mb = round(sum(content_bytes) / 1048576.0, decimals: 2)}
| fieldsAdd retention_35d_gb = round(daily_mb * 35 / 1024, decimals: 2)
| fieldsAdd retention_7d_gb = round(daily_mb * 7 / 1024, decimals: 2)
| fieldsAdd savings_gb = round(retention_35d_gb - retention_7d_gb, decimals:
2)
| fieldsAdd savings_pct = round((savings_gb * 100) / retention_35d_gb,
decimals: 1)
```

```python
// Retention planning: volume by bucket and loglevel
fetch logs, from: now() - 24h
| fieldsAdd content_bytes = stringLength(content)
| summarize {
    log_count = count(),
    daily_mb = round(sum(content_bytes) / 1048576.0, decimals: 2)
  }, by: {dt.system.bucket, loglevel}
| sort dt.system.bucket asc, daily_mb desc
```
~~~

~~~
## 4. Bucket Routing Configuration

Route logs to appropriate buckets using **OpenPipeline routing rules**.

### OpenPipeline Routing Rules

```yaml
# Route logs based on conditions
routes:
  # Route debug logs to short-retention bucket
  - name: debug-to-debug-bucket
    condition: loglevel == "DEBUG" OR loglevel == "TRACE"
    bucket: debug_logs

  # Route errors to extended-retention bucket
  - name: errors-to-error-bucket
    condition: loglevel == "ERROR" OR loglevel == "FATAL"
    bucket: error_logs

  # Route audit logs to compliance bucket
  - name: audit-to-audit-bucket
    condition: contains(content, "audit") OR contains(k8s.namespace.name,
"security")
    bucket: audit_logs

  # Default catch-all
  - name: default-route
    condition: true
    bucket: default_logs
```

### Routing Priority

1. First matching rule wins
2. Order rules from specific to general
3. Always include a default catch-all

```python
// Preview routing decision logic
fetch logs, from: now() - 1h
| limit 1000
| 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 target_bucket asc, count desc
```

```python
// Validate current routing effectiveness
fetch logs, from: now() - 24h
| summarize {
    total = count(),
    in_default = countIf(dt.system.bucket == "default_logs")
  }, by: {loglevel}
| fieldsAdd default_pct = round((in_default * 100.0) / total, decimals: 1)
| sort default_pct desc
```

```python
// Namespace-based routing analysis
fetch logs, from: now() - 24h
| filter isNotNull(k8s.namespace.name)
| fieldsAdd suggested_bucket = if(contains(k8s.namespace.name, "prod"),
"prod_logs",
                               else: if(contains(k8s.namespace.name,
"staging"), "staging_logs",
                               else: "dev_logs"))
| summarize {count = count()}, by: {k8s.namespace.name, suggested_bucket}
| sort count desc
| limit 20
```

## 5. Access Control & Data Governance

### Bucket-Level Access Control

| Bucket | Read Access | Write Access | Use Case |
|--------|-------------|--------------|----------|
| `default_logs` | All authenticated | OpenPipeline | General access |
| `debug_logs` | Dev teams | OpenPipeline | Development |
| `audit_logs` | Audit team only | OpenPipeline | Compliance |
| `security_logs` | SecOps only | OpenPipeline | Security ops |

### IAM Policy Integration

Configure in **Settings → Access tokens & IAM → Policies**

```json
{
  "name": "audit-log-readers",
~~~

~~~~
  "statement": [
    {
      "effect": "ALLOW",
      "permissions": ["storage:logs:read"],
      "conditions": [
        {
          "bucket": ["audit_logs"]
        }
      ]
    }
  ]
}
```

```python
// Identify sensitive data patterns for access control planning
fetch logs, from: now() - 24h
| filter contains(content, "password")
        OR contains(content, "secret")
        OR contains(content, "token")
        OR contains(content, "api_key")
        OR contains(content, "credential")
| summarize {count = count()}, by: {dt.system.bucket, k8s.namespace.name}
| sort count desc
| limit 15
```

```python
// Data governance audit: bucket access patterns
fetch logs, from: now() - 24h
| summarize {
    total_logs = count(),
    unique_sources = countDistinct(dt.openpipeline.source)
  }, by: {dt.system.bucket}
| sort total_logs desc
```

## 6. Query Optimization with Buckets

### Target Specific Buckets

Querying specific buckets **reduces scan costs** and **improves
performance**.

**✅ Efficient - targets specific bucket:**
```
fetch logs, bucket: "error_logs", from: now() - 7d
| filter loglevel == "ERROR"
~~~~

~~~~
| limit 100
```

**❌ Inefficient - scans all buckets:**
```
fetch logs, from: now() - 7d
| filter loglevel == "ERROR"
| limit 100
```

### Multi-Bucket Queries

```
// Query multiple specific buckets
fetch logs, bucket: {"prod_logs", "error_logs"}, from: now() - 24h
| filter loglevel == "ERROR"
```

```python
// Query specific bucket (efficient)
fetch logs, bucket: "default_logs", from: now() - 1h
| summarize {count = count()}, by: {loglevel}
| sort count desc
```

```python
// Compare bucket query efficiency
fetch logs, from: now() - 1h
| summarize {
    total_logs = count(),
    buckets_scanned = countDistinct(dt.system.bucket)
  }
```

```python
// Bucket-aware dashboarding: hourly volume by bucket
fetch logs, from: now() - 24h
| makeTimeseries {count = count()}, by: {dt.system.bucket}, interval: 1h
| limit 100
```

## 7. Cost Optimization Strategies

### Storage Cost Reduction Tactics

| Strategy | Savings | Implementation |
|----------|---------|----------------|
| Shorten DEBUG retention | 60-80% | Route to 7-day bucket |
~~~~

~~~
| Drop health check logs | 10-30% | OpenPipeline filter |
| Sample verbose logs | 50-90% | Sampling processor |
| Compress long content | 20-40% | Field truncation |

### ROI Calculation

```
Current State:
- 100 GB/day total logs
- 35-day retention all buckets
- Total storage: 3,500 GB

Optimized State:
- 60 GB/day INFO/WARN (35d) = 2,100 GB
- 30 GB/day DEBUG (7d) = 210 GB
- 10 GB/day ERROR (90d) = 900 GB
- Total storage: 3,210 GB

Savings: ~8% immediate, scales with volume
```

```python
// Calculate potential cost savings
fetch logs, from: now() - 24h
| fieldsAdd content_bytes = stringLength(content)
| summarize {
    daily_gb = round(sum(content_bytes) / 1073741824.0, decimals: 2),
    log_count = count()
  }, by: {loglevel}
| fieldsAdd current_35d_gb = round(daily_gb * 35, decimals: 1)
| fieldsAdd optimal_retention = if(loglevel == "DEBUG" OR loglevel ==
"TRACE", 7,
                                else: if(loglevel == "ERROR" OR loglevel ==
"FATAL", 90,
                                else: 35))
| fieldsAdd optimal_storage_gb = round(daily_gb * optimal_retention,
decimals: 1)
| fieldsAdd savings_gb = round(current_35d_gb - optimal_storage_gb, decimals:
1)
| sort savings_gb desc
```

```python
// Identify high-volume low-value logs
fetch logs, from: now() - 24h
| filter loglevel == "DEBUG" OR loglevel == "INFO"
| fieldsAdd content_bytes = stringLength(content)
| summarize {
~~~

~~~
    log_count = count(),
    total_mb = round(sum(content_bytes) / 1048576.0, decimals: 2)
  }, by: {k8s.namespace.name, k8s.workload.name}
| sort total_mb desc
| limit 20
```

## 8. Bucket Management Best Practices

### ✅ Do's

1. **Start with clear bucket strategy** before configuring pipelines
2. **Name buckets descriptively** with purpose and team
3. **Set retention based on data value**, not convenience
4. **Review bucket distribution** weekly
5. **Query specific buckets** for performance
6. **Document bucket purposes** and access policies

### ❌ Don'ts

1. **Don't put everything in default bucket** - defeats purpose
2. **Don't set uniform retention** - wastes storage on low-value data
3. **Don't ignore bucket metrics** - monitor distribution
4. **Don't create too many buckets** - complexity overhead
5. **Don't skip access controls** - security risk

```python
// Weekly bucket health check query
fetch logs, from: now() - 7d
| summarize {
    total_logs = count(),
    unique_sources = countDistinct(dt.openpipeline.source),
    debug_pct = round((countIf(loglevel == "DEBUG") * 100.0) / count(),
decimals: 1),
    error_pct = round((countIf(loglevel == "ERROR") * 100.0) / count(),
decimals: 1)
  }, by: {dt.system.bucket}
| sort total_logs desc
```

```python
// Bucket routing effectiveness
fetch logs, from: now() - 24h
| summarize {count = count()}, by: {dt.system.bucket,
dt.openpipeline.pipelines}
| sort count desc
```
~~~

```
---

## 📝 Summary

In this notebook, you learned:

✅ **Bucket Architecture** - How Grail organizes log storage
✅ **Strategic Design** - Creating purpose-driven buckets
✅ **Retention Policies** - Optimizing data lifecycle costs
✅ **Routing Rules** - Directing logs to appropriate storage
✅ **Access Control** - Implementing data governance
✅ **Query Optimization** - Targeting buckets for efficiency

### Key Takeaway

> **Buckets are foundational.** Design your bucket strategy first - it
impacts security, cost, and every query you'll write.

---

## ➡️ Next Steps

Continue to **OPLOGS-05: Querying & Parsing** to learn advanced DQL
techniques for log analysis.

---

## 📚 References

- [Grail Buckets](https://docs.dynatrace.com/docs/platform/grail/data-
model/buckets)
- [Log Storage Configuration](https://docs.dynatrace.com/docs/manage/data-
privacy-and-security/data-management/log-monitoring-data-storage)
- [Data Retention](https://docs.dynatrace.com/docs/manage/data-privacy-and-
security/data-management/data-retention)
- [IAM Policies](https://docs.dynatrace.com/docs/manage/identity-access-
management/permission-management/manage-user-permissions-policies)
```