

corewire

Rook-Ceph
Administration

Folien-Hinweis

- `Space`, `Page down`: Nächste Folie
- `Page up`: Vorherige Folie
- `ESC`, `o`: Übersicht

[Zur Kapitelübersicht](#)

Ziele dieses Kapitels

Ich kann

- ✓: Den Zustand eines Rook-Ceph-Clusters anhand zentraler
- ✓: Signale beurteilen
- ✓: Typische Alltagsaufgaben im Betrieb einordnen
- ✓: Eine geplante Node-Wartung sicher durchführen
- Einen defekten Datenträger kontrolliert austauschen

Clusterzustand beurteilen

Cluster health interpretieren

- `HEALTH_OK`: Cluster arbeitet vollständig wie erwartet
- `HEALTH_WARN`: Cluster funktioniert, aber es gibt etwas zu prüfen
- `HEALTH_ERR`: Cluster ist in einem kritischen Zustand, Eingreifen erforderlich

Konkrete Ursache einer Warnung/eines Fehlers

```
ceph health detail
```

MON-Quorum prüfen

- MONs müssen sich auf einen gemeinsamen Cluster-Zustand einigen (Quorum)
- Ohne Quorum sind keine Schreib-/Lesevorgänge im Cluster möglich
- Quorum benötigt eine Mehrheit der konfigurierten MONs

Anzahl und Zustand der MONs im Quorum

```
ceph mon stat
```

OSD-Zustände: `up/down` und `in/out`

- `up/down`: Ist der OSD-Daemon aktuell erreichbar und läuft?
- `in/out`: Ist der OSD Teil der Datenverteilung (CRUSH)?
- Ein OSD kann `up` und gleichzeitig `out` sein (bewusst herausgenommen)
- Ein OSD kann `down` und trotzdem noch `in` sein (Daten werden ihm weiter zugeordnet)

Placement-Group-Zustände interpretieren

- `active+clean`: Zielzustand, alle Kopien vorhanden und synchron
- `degraded`: mindestens eine Kopie einer PG fehlt
- `undersized`: weniger Kopien vorhanden als von der Pool-Konfiguration gefordert
- `peering`: OSDs stimmen sich gerade über den Zustand einer PG ab
- `recovering` / `backfilling`: PG wird gerade wiederhergestellt bzw. neu verteilt

Kapazität und Verteilung beurteilen

Gesamt- und Pool-Auslastung

```
ceph df
```

Auslastung einzelner OSDs

```
ceph osd df
```

- Ungleichmäßige Verteilung führt dazu, dass einzelne OSDs früher **NEARFULL** werden
- Wichtig: Nicht nur Gesamtkapazität, sondern auch Verteilung über OSDs beachten

Ungleichmäßige Verteilung – Ursachen

- Unterschiedliche Disk-Größen innerhalb desselben Pools/Hosts
- Zu wenige Placement Groups für die vorhandene Datenmenge
- CRUSH-Regeln oder Failure Domains, die Daten ungünstig verteilen
- Einzelne sehr große RBD-Images oder Objekte („Hotspots“)
- Kürzlich hinzugefügte OSDs, die noch nicht vollständig aufgefüllt sind

Slow Operations erkennen

- Ceph meldet Operationen, die ungewöhnlich lange dauern, als „slow ops“
- Häufige Ursachen: überlastete Disks, Netzwerkprobleme, überlastete OSDs
- Sichtbar in `ceph status` und `ceph health detail`
- Slow Ops sind oft ein Frühwarnsignal, bevor ein Problem eskaliert

Recovery, Backfill und Rebalancing überwachen

- Nach Ausfällen oder Konfigurationsänderungen verschiebt Ceph Daten neu
- `ceph status` zeigt den Fortschritt laufender Recovery-/Backfill-Vorgänge
- Hohe Recovery-Last kann reguläre Client-I/O spürbar beeinträchtigen
- Ziel: Fortschritt beobachten, ohne den Produktivbetrieb zu gefährden

Typische Alltagsaufgaben

Was gehört zum Tagesgeschäft?

- Neue Storage-Kapazität hinzufügen
- Ressourcenverbrauch der Ceph-Daemons untersuchen
- Ungleichmäßige Speicherverteilung erkennen
- Near-Full- und Full-Zustände vermeiden
- **Defekte Datenträger austauschen**
- **Kubernetes-Nodes warten**

Flags

- Ceph-Flags schalten bestimmtes Cluster-Verhalten gezielt ein oder aus
- `noout`: OSDs werden bei Nichterreichbarkeit nicht automatisch `out` gesetzt
- `noscrub` / `nodeep-scrub`: Scrubbing wird vorübergehend pausiert
- `norebalance` / `norecover`: Rebalancing bzw. Recovery wird vorübergehend gestoppt
- Sinnvoll für geplante, zeitlich begrenzte Eingriffe am Cluster
- Gesetzte Flags müssen nach dem Eingriff wieder entfernt werden, sonst bleibt der Schutz dauerhaft aktiv

Geplante Wartung

Kubernetes-Node sicher warten

1. Clusterzustand als Referenz prüfen (`ceph status`)
2. Node kontrolliert drainen, bevor er neu gestartet wird
3. Node wieder kontrolliert in den Cluster aufnehmen
4. Zustand mit der Referenz vor der Wartung vergleichen

Ausgangszustand prüfen

Bevor du einen Node cordonst, vergewisserst du dich, dass der Cluster gesund ist.

```
ceph status
```

Achte auf:

```
health: HEALTH_OK  
...  
pgs: active+clean
```

Pod Disruption Management

```
kubectl -n rook-ceph get cephcluster rook-ceph \
-o jsonpath='{.spec.disruptionManagement}'
```

```
spec:
  disruptionManagement:
    managePodBudgets: true
    osdMaintenanceTimeout: 30
```

Was `managePodBudgets` bedeutet

- Rook legt selbst `PodDisruptionBudgets` für die OSDs an
- Bei einem Drain setzt Rook automatisch das Flag `noout`
- OSDs werden während der Wartung nicht sofort entfernt
- Nach der Wartung oder nach `osdMaintenanceTimeout` entfernt Rook `noout` wieder
- Konsequenz: `noout` muss bei einem regulären `kubectl drain` **nicht manuell** gesetzt werden

Node cordonen

Node als nicht mehr schedulable markieren, damit keine neuen Pods dort landen.

```
kubectl cordon <NODE>
```

Node drainen

```
kubectl drain <NODE> --ignore-daemonsets --delete-emptydir-data
```

- OSD-Pods sind über Node-Affinity fest an ihren Node gebunden
- Sie werden deshalb nicht verschoben, sondern beendet und bleiben `Pending`
- Ceph markiert die betroffenen OSDs als `down`, sobald ihr Heartbeat ausbleibt

Node wieder einbinden

```
kubectl uncordon <NODE>
```

Zustand nach der Wartung validieren

- Prüfen, ob der Node wieder `Ready` ist
- Prüfen, ob alle Rook-Ceph-Pods wieder laufen
- Prüfen, ob alle erwarteten OSDs wieder `up` und `in` sind
- Prüfen, ob der Cluster wieder `HEALTH_OK` meldet
- Prüfen, ob keine Recovery mehr aktiv ist

Node Status prüfen

```
kubectl get node <NODE>
```

```
kubectl -n rook-ceph get pods -l app=rook-ceph-osd -o wide
```

```
ceph status
```

Achte erneut auf `health: HEALTH_OK` und `pgs: active+clean`.

`in/out` in `ceph osd tree` ablesen

```
ceph osd tree
```

ID	CLASS	WEIGHT	TYPE NAME	STATUS	REWEIGHT	PRI-AFF
2	hdd	1.00000	osd.2	up	1.00000	1.00000

- STATUS: `up` oder `down` (Daemon-Zustand)
- REWEIGHT: `1.00000` bedeutet `in`, `0` bedeutet `out`

Defekte Datenträger austauschen

Zwei Varianten für den Austausch

- **Klassische Variante:** OSD komplett entfernen
 - Neue OSD → **neue OSD-ID und CRUSH-Position**
 - Neue OSD → Rebalancing über den ganzen Cluster
- **Neue Variante:** OSD-Deployment wird annotiert
 - Datenträger wird getauscht
 - OSD-ID und CRUSH-Position bleiben erhalten
 - Ceph backfillt nur die Daten dieser einen OSD
 - kein clusterweites Rebalancing

Zwei Varianten für den Austausch

- Die neue Variante ist zum Zeitpunkt dieser Schulung noch nicht in einem offiziellen Rook-Release enthalten, sondern nur in der Entwicklungsdokumentation
- Wir nutzen sie in diesem Kapitel trotzdem, weil sie der deutlich präzisere und schonendere Ansatz ist

Ablauf beim Datenträgertausch

1. Defekten Datenträger anhand von OSD- und Node-Signalen identifizieren
2. Annotation am OSD-Deployment setzen
3. Auf die Bereitschaft zum Tausch warten
4. Datenträger austauschen
5. Ceph backfillt die Daten auf die neue OSD zurück, bis alle PGs wieder `active+clean` sind

Defekten Datenträger identifizieren

- `ceph health detail` / `ceph status`: OSD wird als `down` oder mit Fehlern gemeldet
- `ceph osd tree`: zeigt, welche OSD betroffen ist und auf welchem Host sie läuft

Gerätenamen z. B. `sdb` und einen Pfad z. B. `/dev/disk/by-path/...`

```
ceph osd metadata <OSD-ID> | grep -E "\"devices\"|\"device_paths\""
```

Austausch über Annotation

- Austausch über eine Annotation am OSD-Deployment
- Nur der Datenträger wird getauscht
- Die OSD-ID und CRUSH-Position bleiben erhalten
- Ceph backfillt nur die Daten dieser einen OSD

Austausch der OSD auslösen

```
kubectl -n rook-ceph annotate deployment rook-ceph-osd-<OSD-ID> \
  osd.rook.io/replace=yes-really-replace-osd-<OSD-ID>
```

Auf die Bereitschaft zum Tausch warten

```
kubectl -n rook-ceph get deployment rook-ceph-osd-<OSD-ID> \
-o jsonpath='{.metadata.annotations.osd\.rook\.io/replace-ready-for-swap}{"\n"}'
```

- Solange der Wert leer ist, drained die OSD noch
- Sobald die Annotation `osd.rook.io/replace-ready-for-swap` gesetzt ist, hat Rook die OSD gedraint und zerstört
- Parallel beobachtbar mit `ceph osd df osd.<OSD-ID>`: Auslastung sinkt gegen 0

Datenträger tauschen

Hier wird jetzt die Platte getauscht.

Austausch verifizieren

```
kubectl -n rook-ceph get deployment rook-ceph-osd-<OSD-ID>
```

- Deployment sollte wieder `1 / 1` `READY` melden, mit derselben OSD-ID wie vor dem Austausch

```
ceph status
```

```
ceph osd df osd.<OSD-ID>
```

- Ceph backfillt die Daten auf die neue OSD zurück, bis alle PGs wieder `active+clean` sind

Administrationswerkzeuge

Werkzeuge für den Alltag

- Kubernetes Events und Logs für die Kubernetes-Sicht auf Rook/Ceph-Pods
- Ceph Dashboard für eine grafische Sicht auf Clusterzustand und Metriken
- Rook-Toolbox für direkte Ceph-Kommandos

Wichtige Ceph-Kommandos

Befehl	Zweck
<code>ceph health detail</code>	Ursache von <code>HEALTH_WARN</code> / <code>HEALTH_ERR</code> anzeigen
<code>ceph df</code>	Kapazität von Cluster und Pools
<code>ceph osd status</code>	Kompakte Übersicht über alle OSDs
<code>ceph osd tree</code>	OSDs nach Host/CRUSH-Hierarchie

Wichtige Ceph-Kommandos

Befehl	Zweck
<code>ceph osd df</code>	Auslastung einzelner OSDs
<code>ceph pg stat</code>	Zustand der Placement Groups
<code>ceph mon stat</code>	Zustand und Quorum der MONs

Lab

- Gehen Sie auf: <https://labs.corewire.de>
- Navigieren Sie nach:
 - Rook
 - Aufgaben
 - Administration

Zur Kapitelübersicht

- Vorheriges Kapitel: [Konfiguration](#)