Proxmox - Ceph

Ceph (Aussprache /'sɛf/) ist eine quelloffene verteilte Speicherlösung (Storage-Lösung) in der Informationstechnik. Kernkomponente ist mit RADOS (englisch reliable autonomic distributed object store) ein über beliebig viele Server redundant verteilbarer Objektspeicher (englisch object store). Ceph bietet dem Nutzer drei Arten von Storage an: Einen mit der Swift- und S3-API kompatiblen Objektspeicher (RADOS Gateway), virtuelle Blockgeräte (RADOS Block Devices) und CephFS, ein verteiltes Dateisystem.[4]

Ceph kann als RADOS Block Device (RBD) über das Ceph iSCSI Gateway auch als hochverfügbares iSCSI-Target bereitgestellt werden. Dadurch kann es auf Client-Seite durch viele Betriebssysteme (auch Windows) genutzt werden.

- Crushmap dekompilieren / kompilieren
- Pools nach Classen anlegen (SSD Pool / HDD Pool)
- Nach löschen eines Ceph Monitors bleibt immer noch der eintrag über
- Nach entfernen eines Hosts bleibt in der Chrush Map der leere eintrag übrig

Crushmap dekompilieren / kompilieren

Beschreibung:

Die Ceph Crushmap ist eine Konfigurationsdatei in Ceph, die beschreibt, wie Daten auf physische Speicherressourcen verteilt werden sollen. Sie definiert die Hierarchie der Speichergeräte und ihre Beziehungen zueinander, um die Ausfallsicherheit und Leistung von Ceph-Clustern zu optimieren. Die Crushmap wird von der Crush-Algorithmus-Engine verwendet, um intelligente Verteilungsentscheidungen zu treffen und sicherzustellen, dass Daten auch bei Ausfällen von Speichergeräten zugänglich bleiben.

Dekompilieren

Crushmap im Binärformat speichern.

Syntax

```
ceph osd getcrushmap -o <zieldatei>
ceph osd getcrushmap -o ~/crush_map_compressed_2023-03-29
```

Nun die Binärdatei dekompilieren. Somit haben wir auch gleichzeitig eine Sicherungskopie, nämlich die Kompilierte Datei

Sytntax

```
crushtool -d <binärcrushmapdatei> -o <zielcrushmapdateidecompiled>


crushtool -d ~/crush_map_compressed_2023-03-29 -o ~/crush_map_decompiled_2023-03-29
```

Nun kann die crushdatei mit einem ganz normelen Texteditor wie vi,vim,nano,joe etc. bearbeitet werden.

Kompilieren und wieder injekzieren

```
crushtool -c <crushmapdateidecompiled> -o <neuebinärcrushmapdatei>
crushtool -c ~/crush_map_decompiled_2023-03-29 -o ~/new_crush_map_compressed_2023-03-29
```

Nun kann diese Datei wieder Injekziert werden

Syntax

ceph osd setcrushmap -i <neuebinärcrushmap>

ceph osd setcrushmap -i ~/new_crush_map_compressed_2023-03-29

Fertig.

Pools nach Classen anlegen (SSD Pool / HDD Pool)

Beschreibung:

In Ceph können Sie verschiedene Pools anlegen, um unterschiedliche Typen von Speichergeräten zu nutzen. Zum Beispiel können Sie SSD-, NVME- und HDD-Pools erstellen, um eine optimale Nutzung der verfügbaren Speichergeräte zu erreichen. Jeder Pool kann unterschiedliche Konfigurationen für die Leistung und die Ausfallsicherheit aufweisen, um den Anforderungen der Anwendung gerecht zu werden. Durch die Verteilung von Daten auf verschiedene Speichergeräte können Sie auch sicherstellen, dass Daten bei Ausfällen von Speichergeräten zugänglich bleiben.

Festplatten Typen (Classes)

In Ceph können einer OSD verschiedene Klassen hinzugefügt werden.
Diese sind:

1. ssd: Eine OSD-Klasse für SSD-Speichergeräte. Diese OSDs werden normalerweise für Daten verwendet, die eine höhere Leistung erfordern.
2. hdd: Eine OSD-Klasse für HDD-Speichergeräte. Diese OSDs werden normalerweise für Daten verwendet, die keine hohe Leistung erfordern, sondern eher für Archiv- oder Backupzwecke geeignet sind.
3. nvme: Eine OSD-Klasse für NVMe-Speichergeräte. Diese OSDs werden normalerweise für Daten verwendet, die eine sehr hohe Leistung erfordern, z.B. für Anwendungen mit hohem I/O-Durchsatz.

Festplatten Klasse löschen/setzen

`Ist schon eine Festplattenklasse gesetzt muss diese erst gelöscht werden, bevor eine neue vergeben werden kann. Im Terminal dienen folgende Befehle zum löschen/setzen. In unserem Fall die OSD.12 auf NVME setzten.`

`Löschen`

```
ceph osd crush rm-device-class <osdnr>

ceph osd crush rm-device-class osd.12

Ausgabe:

done removing class of osd(s): 12
```

Setzen

Syntax

classtype = hdd,ssd,nvme

ceph osd crush set-device-class <classtype> <osdnr>

ceph osd crush set-device-class nvme osd.12

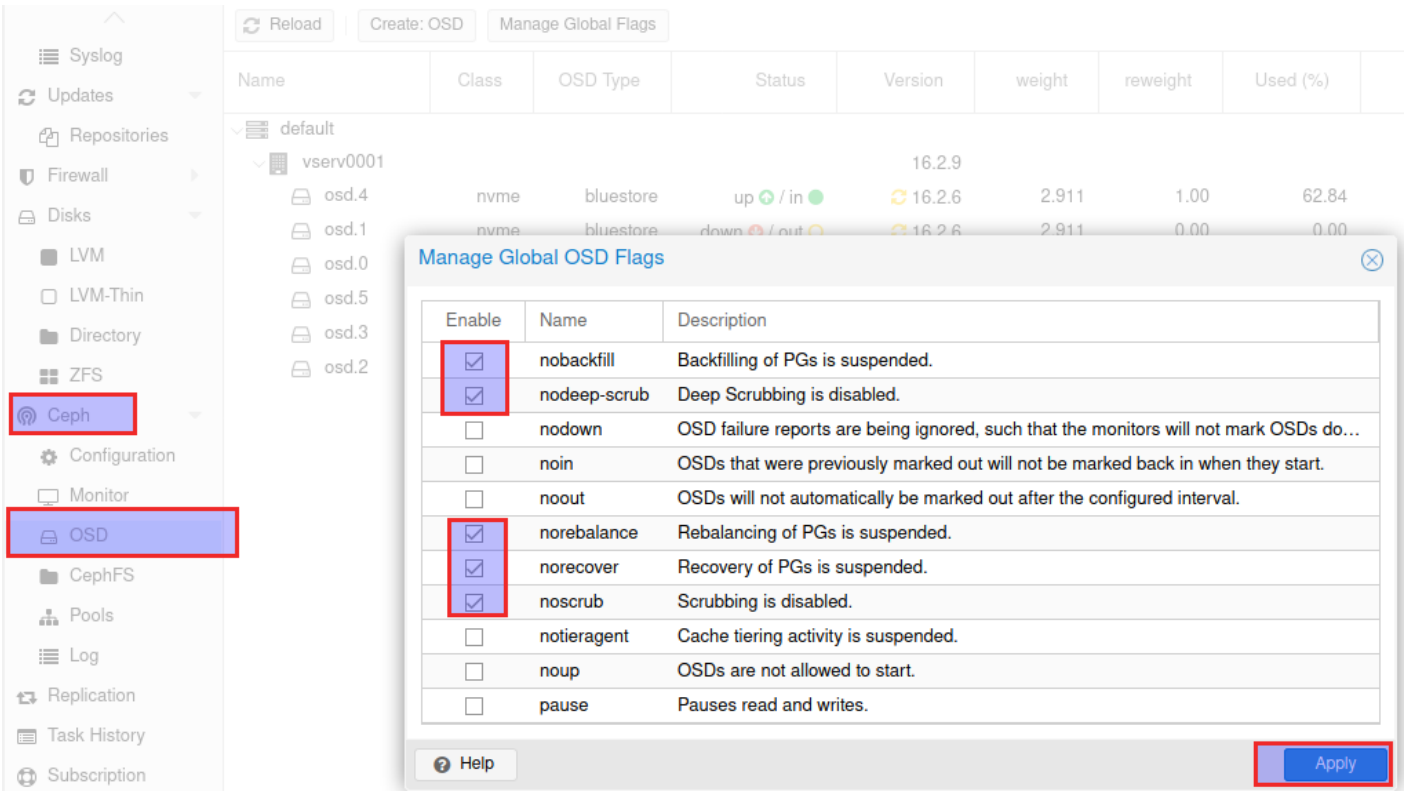
Ausgabe:

set osd(s) 12 to class 'nvme'

Crushmap um weitere replicated rule erweitern/ändern.

!Wichtige Hinweise!!!

!!! WICHTIG!!! Es muss schon eine OSD mit der Klasse geben, bevor eine Rule dafür gebaut werden kann !!!
Ist dies nicht der Fall müsen wir erst die erste Rule nur anpassen für die Klasse die schon da ist.
Danach legen wir dei OSD mit der neuen Klasse an, aber vor alle dem alles auf Maintenance setzten!!!
!!! WICHTIG!!! Alles auf Maintenance stzten, wie no backfill, noreblance etc setzen.



Erst nach erstellung der Rule die haken wieder entfernen.

Über manuelles bearbeiten

Die Crushmap decompilieren so, das man diese auch bearbeiten kann. Siehe Seite : Crushmap dekompilieren / kompilieren

Ist die Map decompiliert. Der ersten Rule den Parameter für die Klassenzueweisung hinzufügen. Damit wäre dann erste Regel definiert, das diese für HDDs gillt. Sollten nur NVMEs verbaut sein. Dann natürlich die erste Rule mit NVME setzen und die zweite dann auf HDDs. In unserem Scenario sind aber HDDs vorhanden und wir wollen NVME Pool hinzufügen. Sind noch gar keine Pools definiert, so kann man auch die erste Rule noch umbenennen, z.b noch _hdd hinzufügen. Ist die Rule allerdings schon einem pool zugewiesen, muss dies aktualisiert werden. Was in einer Prod umgebung nicht wirklich Sinnvoll ist, dann einfach als gedanken lassen, die erste Rule ist die Rule von dem Klassentyp die zu erst da war. Wie bei uns die HDDs.

Hier der Parameter der einer Rule hinzugefügt werden muss

```
step take default class <classtype>
step take default class hdd
```

In unsere Crushmap sehe das so aus.
Auszug.

```
Original
...
# rules
rule replicated_rule {
        id 0
        type replicated
        min_size 1
        max_size 10
        step chooseleaf firstn 0 type host
        step emit
}
...

Nun geändert mit HDD class


...
```

```
# rules
rule replicated_rule {
        id 0
        type replicated
        min_size 1
        max_size 10
        step take default class hdd
        step chooseleaf firstn 0 type host
        step emit
}
```

Diese speichern und wieder zurückzuspielen.

Siehe Seite : Crushmap dekompilieren / kompilieren

Nun eine zweite Rule anlegen, dazu die erste kopieren und die ID und Namen ändern.

z.b so

```
...
# rules
rule replicated_rule {
        id 0
        type replicated
        min_size 1
        max_size 10
        step take default class hdd
        step chooseleaf firstn 0 type host
        step emit
}

# rules
rule replicated_rule_nvme {
        id 1
        type replicated
        min_size 1
        max_size 10
        step take default class nvme
        step chooseleaf firstn 0 type host
        step emit
}
```

Diese speichern und wieder zurückzuspielen.

Siehe Seite : Crushmap dekompilieren / kompilieren

Fertig, nun Pool anlegen.

Über Kommandozeilen Befehl

Die Erste Rule die Klasse hinzufügen, wenn die noch keine Klasse hat

```
ceph osd crush rule modify <rulename> set class <classtype>


Beispiel
ceph osd crush rule modify replicated_rule set class hdd
```

Nun wenn nicht schon vorhanden OSD mit neuer Klasse hinzufügen.
Dann die weitere Rule erstellen

Erklärung der Parameter

<rule-name>	Name der Regel, um sie mit einem Pool zu verbinden (angezeigt in der GUI und CLI)
<root>	Zu welchem Crush-Root sie gehören sollte (Standard-Ceph-Root "default")
<failure-domain>	An welchem Fehlerbereich die Objekte verteilt werden sollten (normalerweise Host) (Bei one Node wäre das hier osd)
<class>	Welcher Typ von OSD-Backing-Store verwendet werden soll (e.g., nvme, ssd, hdd)

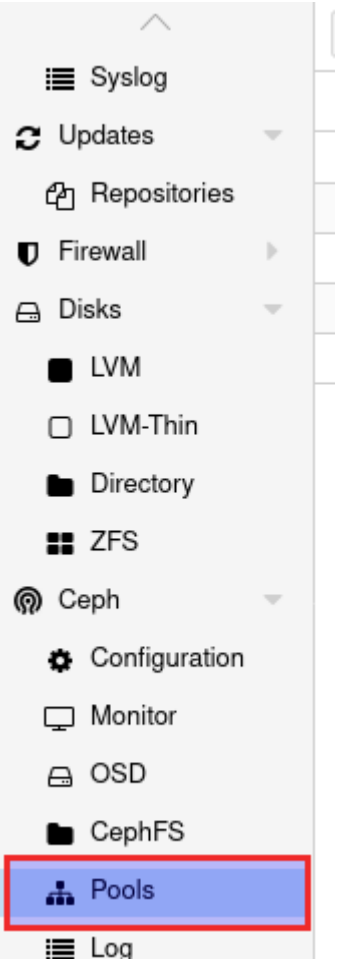
```
Syntax
ceph osd crush rule create-replicated <rule-name> <root> <failure-domain> <class>


Beispiel
ceph osd crush rule create-replicated rule_nvme default host nvme
```

!!! WICHTIG!!! Maintenance mode wieder rausnehmen no backfill, noreblance etc aushaken.

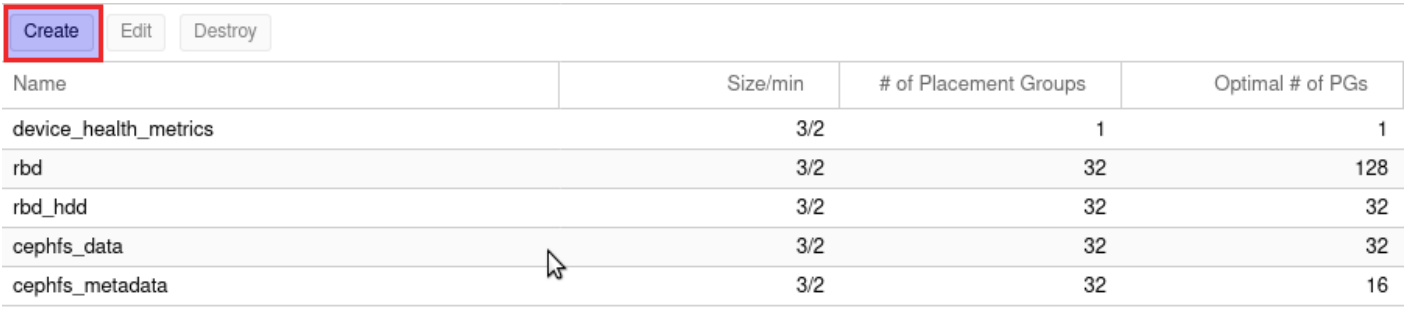
Fertig nun Pool anlegen

Pool anlegen

Nun können wir einen weiteren Pool mit der neuen Rule anlegen.
Dazu auf der Weboberfläche proxmox anmelden, einen Host auswählen dann auf Ceph / Pools klicken

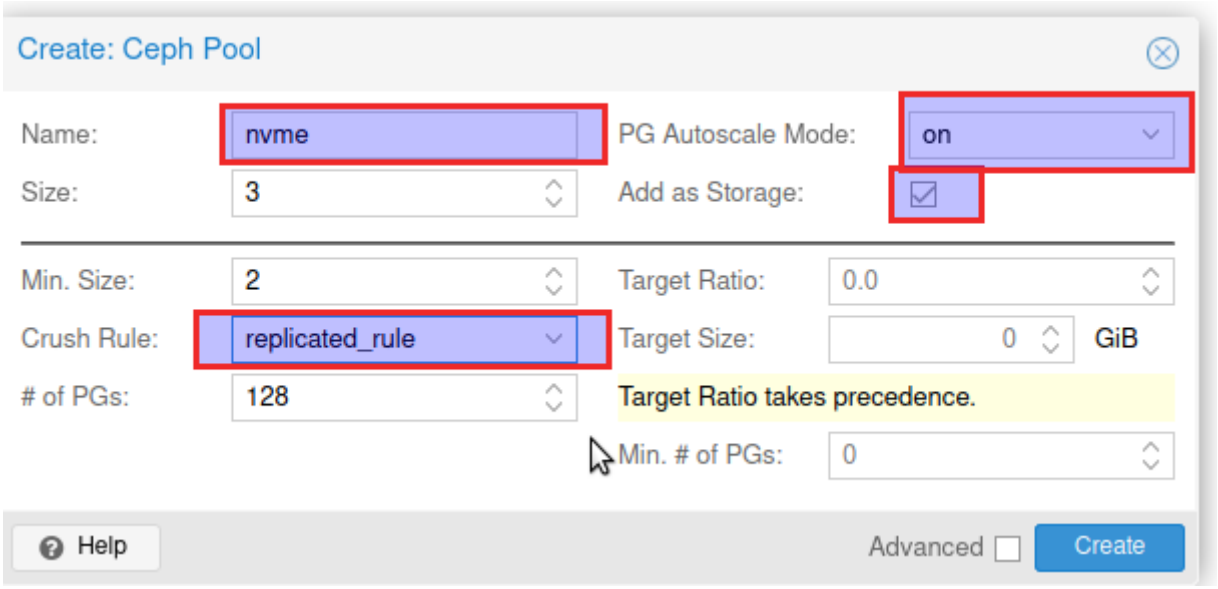


Dann auf Create klicken



Name	Size/min	# of Placement Groups	Optimal # of PGs
device_health_metrics	3/2	1	1
rbd	3/2	32	128
rbd_hdd	3/2	32	32
cephfs_data	3/2	32	32
cephfs_metadata	3/2	32	16

nun einen Namen vergeben und unseer Ruleset auswählen. Auto PG auf on und den Haken bei Add Storage rein.
Dann bekommen wir auch gleich den Speicher zum Datastore um darauf zuzugreifen.



Fertig.

So könnte man zum Beispiel ein RBD Pool für NVMEs bauen und einen CephFS Pool mit HDDs.
Als Beispiel

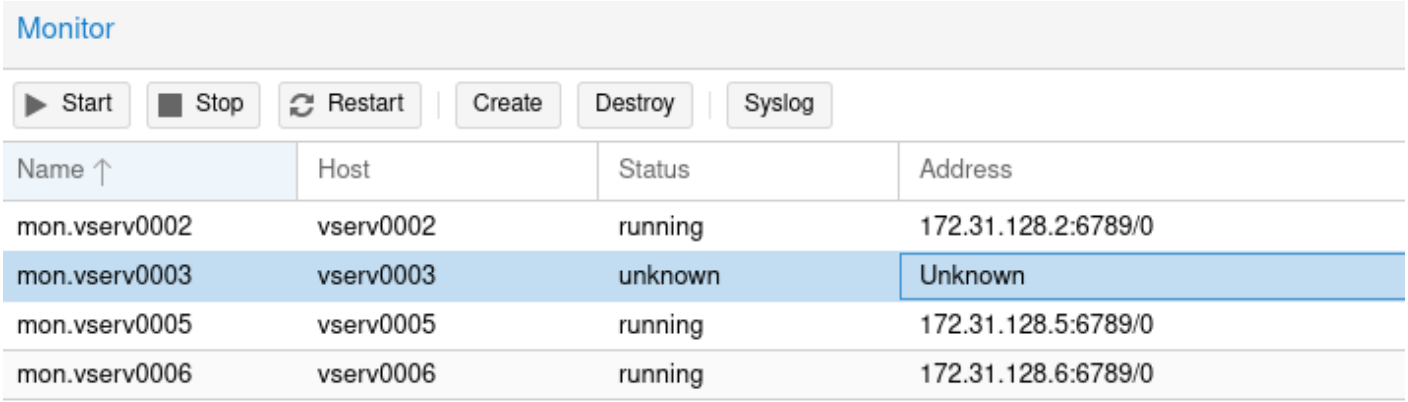
!!!ACHTUNG! NICHT ZU EMPFEHLEN!!!

Man kann natürlich auch einen Vorhandenen Pool ein Komplett anderes Ruleset geben.
Aber Achtung die Daten ziehen dann nicht mit um. Die Änderungen gelten nur für neu geschriebene Daten.
Deshalb lieber einen neuen Pool Anlegen, Daten dort hin verschieben, alten Pool löschen.
Dann einen neuen Pool mit altem Namen und dann der richtigen Ruleset auswählen.
So bleiben die Daten konsistent

Nach löschen eines Ceph Monitors bleibt immer noch der eintrag über

Beschreibung:

Monitor über die GUI gelöscht, bleibt aber als unbekannt stehen.



Lösung 1:

Mittel ceph mon dump schauen ob der eintrag schon raus ist.
Wenn nicht den Monitor nochmals löschen.

```
ceph mon dump
```

Ausgabe, bau uns handelt es sich um mon.vserv0003

```
epoch 13
fsid 0b8471ac-848e-4ceb-8c97-469c327f8390
last_changed 2023-07-31T16:38:00.940392+0200
created 2023-05-14T23:50:35.830167+0200
min_mon_release 17 (quincy)
```

```
election_strategy: 1
0: [v2:172.31.128.2:3300/0,v1:172.31.128.2:6789/0] mon.vserv0002
2: [v2:172.31.128.5:3300/0,v1:172.31.128.3:6789/0] mon.vserv0003
3: [v2:172.31.128.5:3300/0,v1:172.31.128.5:6789/0] mon.vserv0005
4: [v2:172.31.128.6:3300/0,v1:172.31.128.6:6789/0] mon.vserv0006
dumped monmap epoch 13
```

Dann diesen nochmals löschen

```
ceph mon remove <mon-id>
Beispiel:
ceph mon remove vserv0003
```

Lösung 2:

Wenn in der mon dump unser Monitor nicht mehr drin ist, ist noch das Monitor Verzeichnis übrig geblieben.

```
epoch 13
fsid 0b8471ac-848e-4ceb-8c97-469c327f8390
last_changed 2023-07-31T16:38:00.940392+0200
created 2023-05-14T23:50:35.830167+0200
min_mon_release 17 (quincy)
election_strategy: 1
0: [v2:172.31.128.2:3300/0,v1:172.31.128.2:6789/0] mon.vserv0002
1: [v2:172.31.128.5:3300/0,v1:172.31.128.5:6789/0] mon.vserv0005
2: [v2:172.31.128.6:3300/0,v1:172.31.128.6:6789/0] mon.vserv0006
dumped monmap epoch 13
```

Auf dem Host wo der Monitor drauf war einloggen und im Verzeichnis /var/lib/ceph/mon schauen ob das Verzechnis noch da ist.
Wenn ja dieses löschen

```
cd /var/lib/ceph/mon
rm -r ceph-vserv0003/
```

Ergebnis:

Nun ist die Liste wieder korrekt

Monitor

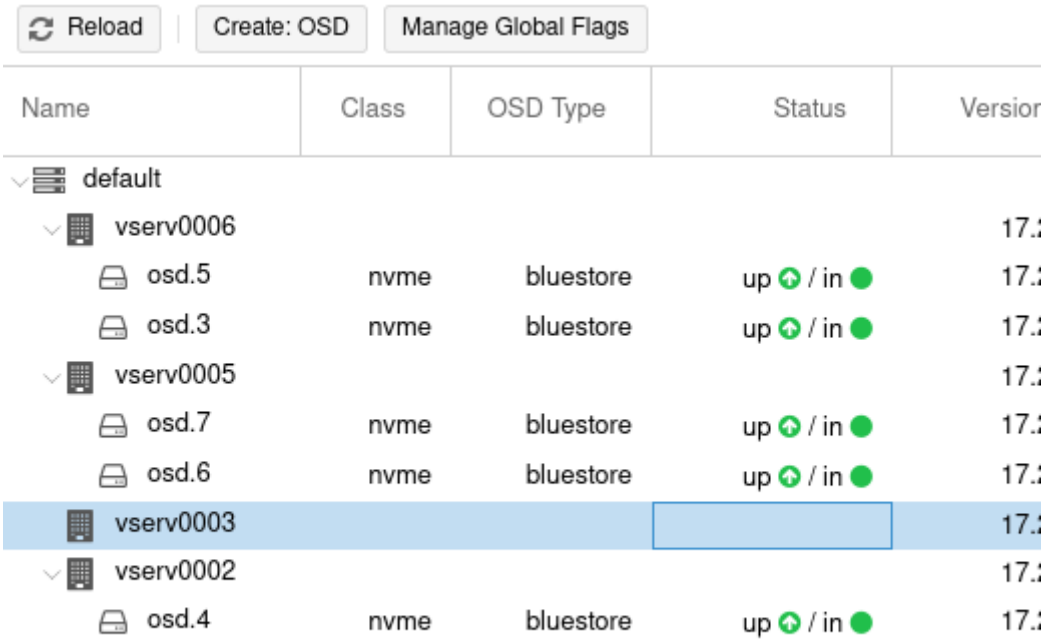
Start | Stop | Restart | Create | Destroy | Syslog

Name ↑	Host	Status	Address
mon.vserv0002	vserv0002	running	172.31.128.2:6789/0
mon.vserv0005	vserv0005	running	172.31.128.5:6789/0
mon.vserv0006	vserv0006	running	172.31.128.6:6789/0

Nach entfernen eines Hosts bleibt in der Chrush Map der leere eintrag übrig

Beschreibung:

Nachdem man die OSD(s) den Monitor den manager , den MDS Manager gelöscht hat.
Bleibt in der Crushmap der Server stehen, halt ohne OSD(s) aber es nervt.



Lösung:

Erstmal schauen ob die Node auch tatsächlich entfernt wurde.
Dieses habem wir vorher mit dem Befehl:

```
pvecm delnode <nodename>
Beispiel:
pvecm delnode vserv0003
```

gemacht.

Um zu sehen das die Node auch wirklich raus ist, rufen wir

```
pvecm nodes
```

Ausgabe:

```
pvecm nodes

Membership information
----------------------
    Nodeid      Votes Name
         1          1 vserv0002
         2          1 vserv0006
         3          1 vserv0005 (local)
```

Die Node 3 ist nicht mehr vorhanden, aber steht trotzdem noch in der Crushmap.
sollte die node doch noch drin stehen dann mit dem obigen befehl nochmals löschen.
Nun können wir den Eintrag aus der Crushmap entfernen.
nun schauen wir welche Nodes in der Crushmap sind

```
ceph osd tree
```

Ausgabe:

```
ceph osd tree
ID   CLASS  WEIGHT    TYPE NAME          STATUS  REWEIGHT  PRI-AFF
 -1        14.55417  root default
 -5         2.91019      host vserv0002
  4   nvme   2.91019          osd.4          up   1.00000  1.00000
 -3               0      host vserv0003
 -9         5.82199      host vserv0005
  6   nvme   2.91100          osd.6          up   1.00000  1.00000
  7   nvme   2.91100          osd.7          up   1.00000  1.00000
-11         5.82199      host vserv0006
  3   nvme   2.91100          osd.3          up   1.00000  1.00000
```

```
 5   nvme   2.91100         osd.5         up   1.00000  1.00000
```

Dies ist die Zeile die uns interessiert

```
...
 -3              0      host vserv0003
...
```

nun den Eintrag löschen

```
ceph osd crush remove {name}
Beispiel:
ceph osd crush remove vserv0003
```

Dann sieht das ganze wieder so aus

Fertig