IOBROKER AUF PROXMOX HA

ÄNDERUNGEN

Datum	Änderungen
01.09.2021	V1.00 - initialer Upload

VORWORT

Inspiriert von den Diskussionen aus

- https://forum.iobroker.net/topic/26327/redis-in-iobroker-%C3%BCberblick
- https://forum.iobroker.net/topic/45979/iobroker-hochverf%C3%BCgbar

ist in den letzten Wochen diese Anleitung entstanden. Vielen Dank an Ingo (@Apollon77) für das Querlesen, Auskunft geben, korrigieren und immer eine freundliche Antwort parat zu haben 😊

ZIEL

Nach dem Durcharbeiten dieser Anleitung steht eine (virtuelle) HA-Umgebung auf Basis von 3 Proxmox-Nodes zur Verfügung. Innerhalb dieser Nodes werden die Daten der Container auf einem Shared-Storage des Typs GlusterFS verfügbar gehalten. ioBroker ist konfiguriert als file(redis)/objects(redis) und hat Zugriff auf 3 Redis-Server-Nodes mit 6 Redis-Server-Prozessen (je LXC einer für die file- und objects-Datenbank). Redis-Sentinel überwacht die Redis-Nodes und stellt die Verfügbarkeit dieser sicher.

Die Anleitung sollte sich für einen produktiven Betrieb auf echter Hardware, z.B. auf ein 3x Intel NUC Setup, adaptieren lassen. Auf der TODO-Liste steht dafür noch die Aktivierung und Beschreibung des Hardware-Watchdogs.

VORRAUSETZUNGEN

- Entstanden ist die Anleitung mit Proxmox in der Version 6.4 und Debian 10 Buster. Tests mit Proxmox 7 und Debian 11 Bullseye stehen noch aus.
- Grundlegendes Verständnis im Umgang mit Linux (hier Debian 10), Proxmox und ioBroker
- Auf das Erstellen der 3 VMs mit Proxmox wird in dieser Anleitung nur oberflächlich eingegangen
- Die Linux Container (LXC) werden aus einem Debian 10 Template erstellt
- Hardware für das Aufsetzen der virtuellen Testumgebung
    - In meinem Beispiel ein aussortierter Intel i7 3770K, 24GB RAM, SSD
- Alle Schritte in dieser Anleitung sind als **root** oder mit vorangestelltem **sudo** auszuführen. Danach ist kein Root Zugriff mehr erforderlich und sollte vermieden werden.
- In den `blauen` Boxen stehen die Shell-Befehle, in den `grünen` die Shell-Ausgaben.

OFFENE PUNKTE & IDEEN FÜR ERWEITERUNGEN

MÖGLICHE ZUKÜNFTIGE ERWEITERUNGEN

- Kapitel: Redis
    - Persistenz: AOF (Append-Only File) anstatt RDB

        - Vorteile, Nachteile, Unterschiede
- Kapitel: Backup erstellen
- Kapitel: Netzwerk erstellen
    - Eigener Switch für die Proxmox Nodes?
    - Eigenes Netz (VLAN) für HA
        - Wenn Intel NUC, wie 2. LAN-Port bereitstellen? USB / Thunderbold?
- Kapitel: Hardware erstellen
    - Intel NUCs im als Cluster
        - 19“ Blende für 3 Intel NUCs: https://www.amazon.de/Rack-Mount-Intel-MiniPC-model/dp/B0886JMZW8
    - Alternativen zum Intel NUC
    - Raspberry PI als Quorum-Device einsetzen (d.h. es werden nur 2 Proxmox-Nodes benötigt)
        - https://pve.proxmox.com/pve-docs/chapter-pvecm.html#_corosync_external_vote_support
- Kapitel: USV erstellen
    - NUT-Server: https://forum.iobroker.net/topic/23688/howto-usv-nut-server-auf-sbc-installieren?_=1621345610029
    - 19“ USV mit geringer Einbautiefe:
        - https://www.cyberpower.com/de/de/product/sku/or600erm1u
        - https://www.cyberpower.com/de/de/product/sku/or1000erm1u

TODOS

- Fertigstellung Befehlsübersichten: kurze Erklärung der Befehle
- Abweichungen bei Produktiv-Einsatz auf echter Hardware (z.B. auf Intel NUCs)
    - Proxmox Cluster / HA und Watchdog
    - https://tompaw.net/proxmox-vm-watchdogs/

```
#/etc/default/pve-ha-manage
# Intel Watchdog
WATCHDOG_MODULE=iTCO_wdt
```

- Kapitel ioBroker
    - Bestehenden ioBroker umstellen (lokale states/objects > redis), inkl. Hinweis auf Multihost Umgebungen: Slaves nicht migrieren!

PROXMOX

CONTAINER UND VMS

Für die Testumgebung werden 3 Virtuelle Maschinen mit einer Proxmox 6.4 Installation und je einer zusätzlichen HDD für das GlusterFS benutzt. Diese Vorgehensweise sollte genauso auch auf 3 Intel NUCs für einen Produktiven Betrieb umzusetzen sein.

Damit Proxmox Virtuelle Maschinen mit wiederum Proxmox bereitstellen kann, muss **Nested Virtualization** aktiviert werden:

```
cat /sys/module/kvm_intel/parameters/nested
```

```
N
```

N bedeutet, das **Nested Virtualization** nicht aktiviert ist.

Das Kernel-Modul **kvm-intel** für Intel CPUs laden:

```
echo "options kvm-intel nested=Y" > /etc/modprobe.d/kvm-intel.conf
modprobe -r kvm_intel
modprobe kvm_intel
reboot
```

Danach gibt der folgende Befehl ein Y aus:

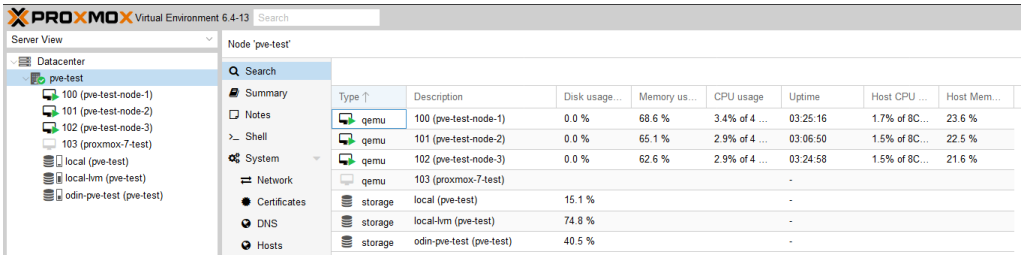
```
cat /sys/module/kvm_intel/parameters/nested
```

```
Y
```

Siehe auch: https://pve.proxmox.com/wiki/Nested_Virtualization

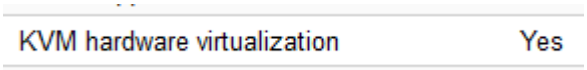
Die 3 VMs innerhalb Proxmox erstellen und grundlegend einrichten:

ID	Name	IP / WebUI
100	pve-test-node-1	192.168.1.61 / https://192.168.1.61:8006
101	pve-test-node-2	192.168.1.62 / https://192.168.1.62:8006
102	pve-test-node-3	192.168.1.63 / https://192.168.1.63:8006



- Benutzer
- Standard-Software: vim, git, curl, ...
- System Updates
- SSH-Zugang

Darauf achten, dass in den Optionen der VMs



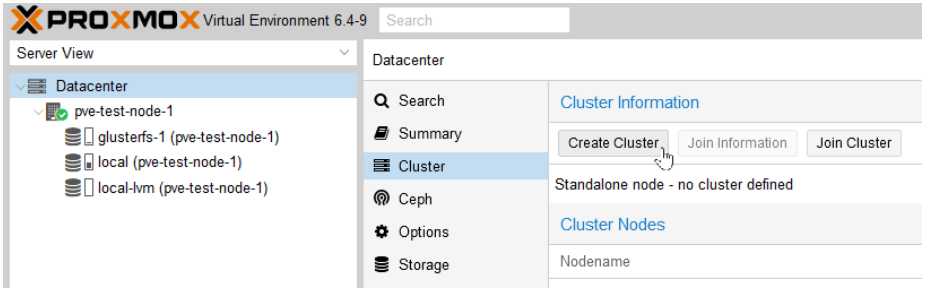
aktiviert ist.

Mit diesen 3 VMs wird im nächsten Kapitel ein Proxmox-Cluster gebildet und darin dann, nach Abschluss des Kapitels „GlusterFS“, die folgenden Linux Container (CT) erstellt:

ID	Name	IP	OS
100	pve-test-iobroker	192.168.1.70	Debian 10
101	pve-test-redis-node-1	192.168.1.71	Debian 10
102	pve-test-redis-node-2	192.168.1.72	Debian 10
103	pve-test-redis-node-3	192.168.1.73	Debian 10

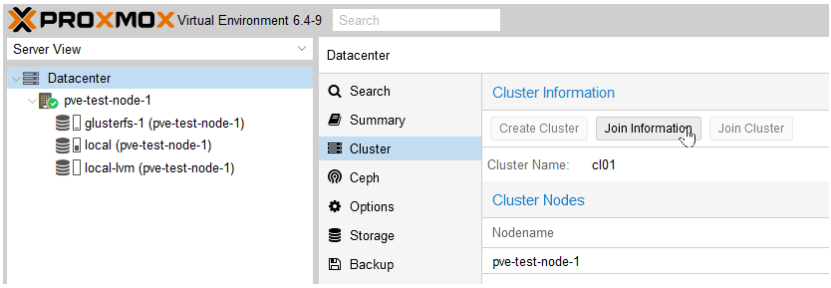
CLUSTER ERSTELLEN

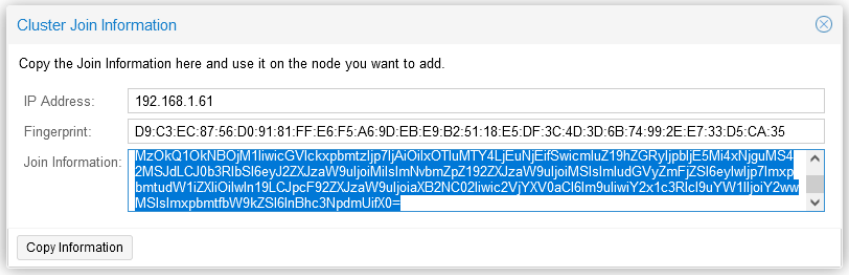
Auf `pve-test-node-1` über **Datacenter > Cluster > Create Cluster** ein Cluster erstellen:



Einen Namen für das Cluster eintragen und mit Create bestätigen:

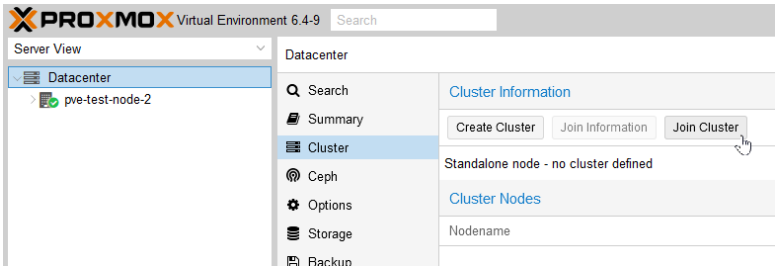
Nach dem anlegen des Clusters über **Join Information** …



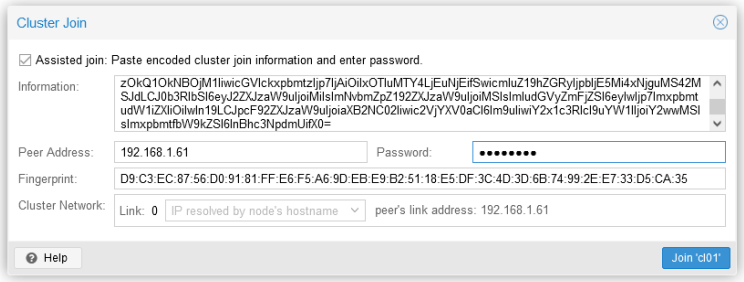


… den Schlüssel für das hinzufügen zusätzlicher Nodes kopieren.

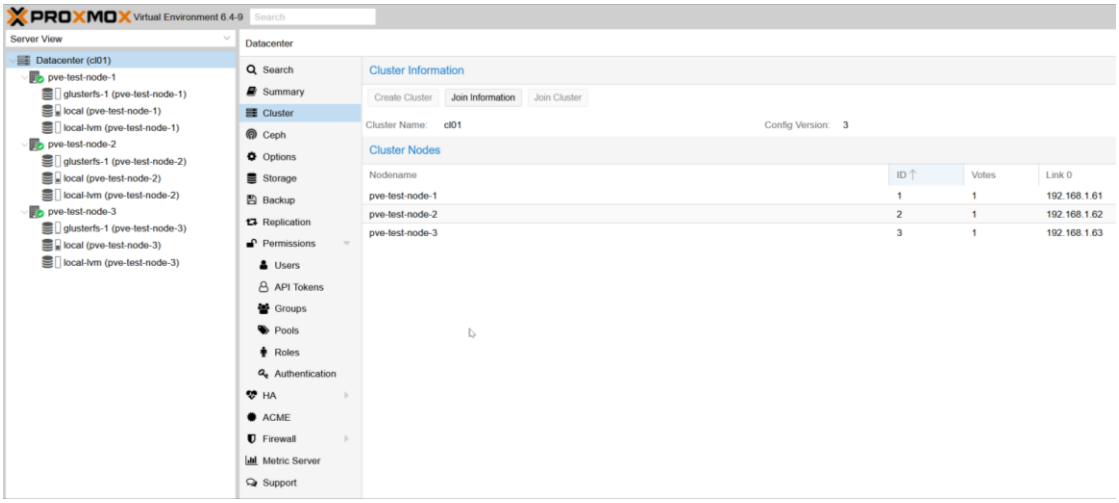
Auf `pve-test-node-2` und `pve-test-node-3` über **Datacenter > Cluster > Join Cluster** dem Cluster CL01 beitreten.



Dazu Schlüssel und Root-Passwort eingeben:



Während dieses Prozesses verlieren die Web-UIs von Node-2 und Node-3 kurz ihre Verbindung. Sobald diese dem Cluster beigetreten sind, können sie wieder normal benutzt werden und zeigen dann jeweils alle Nodes des Clusters an. Damit ist die WebUI des Clusters von allen 3 Nodes erreichbar:



GLUSTERFS ALS SHARED STORAGE

INSTALLATION

TEIL 1 – GLUSTERFS AKTUALISIEREN

Mit Proxmox 6.4 wird eine sehr alte Version von GlusterFS ausgeliefert (Version 5.5). Mit den folgenden Schritten kann die GlusterFS auf 8.x angehoben werden.

Quelle: https://download.gluster.org/pub/gluster/glusterfs/8/LATEST/Debian/

Siehe auch Kapitel „Weiterführendes“ → GLUSTERFS UPGRADE GUIDE V5 > V8

Achtung: Dies muss auf Node 1,2 und 3 ausgeführt werden!

```
wget -O - https://download.gluster.org/pub/gluster/glusterfs/8/rsa.pub | apt-key add -
echo deb [arch=amd64] https://download.gluster.org/pub/gluster/glusterfs/8/LATEST/Debian/buster/amd64/apt buster main >
/etc/apt/sources.list.d/gluster.list
apt update && apt upgrade -y
```

TEIL 2 – GSTATUS INSTALLIEREN

„gstatus is a commandline utility to report the health and other statistics related to a GlusterFS cluster. gstatus consolidates the volume, brick, and peer information of a GlusterFS cluster.

At volume level, gstatus reports detailed infromation on quota usage, snapshots, self-heal and rebalance status“

https://github.com/gluster/gstatus#install

```
curl -LO https://github.com/gluster/gstatus/releases/download/v1.0.6/gstatus
chmod +x ./gstatus
mv ./gstatus /usr/local/bin/gstatus
gstatus --version
```

TEIL 3 – FESTPLATTEN FÜR GLUSTERFS VORBEREITEN

Achtung: Dies muss auf Node 1,2 und 3 ausgeführt werden!

Prüfen, ob die HDD (in unserem Beispiel /dev/sdb) vorhanden ist:

```
fdisk -l /dev/sdb
```

Erwartetes Ergebnis:

```
Disk /dev/sdb: 32 GiB, 34359738368 bytes, 67108864 sectors
Disk model: QEMU HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Ein physisches Volume erstellen:

```
pvcreate /dev/sdb
```

Die dazugehörige „Volume Group“ erstellen:

```
vgcreate vg_glusterfs /dev/sdb
```

Das logical Volume erstellen:

```
lvcreate --name lv_glusterfs -l 100%vg vg_glusterfs
```

Das Volume mit XFS formatieren:

```
mkfs -t xfs -f -i size=512 -n size=8192 -L GlusterFS /dev/vg_glusterfs/lv_glusterfs
```

Das neue Volume ins System einhängen:

```
mkdir -p /data/glusterfs
echo "/dev/mapper/vg_glusterfs-lv_glusterfs /data/glusterfs xfs defaults 0 0" >> /etc/fstab
mount -a
```

GlusterFS installieren:

```
apt install glusterfs-server glusterfs-client -y
```

Dafür sorgen das der glusterd auch nach einem Neustart automatisch gestartet wird:

```
systemctl enable glusterd
```

Danach sicherstellen, dass der GlusterFS Shared Storage Service nicht aktiviert ist, weil dieser die Mount-Reihenfolge beeinflusst und somit beim Herunterfahren das GlusterFS ggf. unmounted wird, wenn noch VMs / LXCs aktiv sind.

```
systemctl status glusterfssharedstorage.service
```

Wenn aktiv, deaktivieren, beenden und den jeweiligen Node neu starten:

```
systemctl disable glusterfssharedstorage.service
systemctl stop glusterfssharedstorage.service
```

Wichtig, am Ende das System neu starten:

```
reboot
```

Grundsätzlich: Wird irgendwas an der GlusterFS-Konfiguration geändert, sollte das System danach durchgestartet werden.

TEIL 4 - GLUSTERFS-VOLUME ERSTELLEN (AUF NODE-1)

Auf `pve-test-node-1` Überprüfen, ob die Verbindung zu `pve-test-node-2` & `pve-test-node-3` hergestellt werden kann:

```
gluster peer probe 192.168.1.62
gluster peer probe 192.168.1.63
```

Erwartetes Ergebnis:

```
peer probe: success
```

Ein GlusterFS Volume über Node-1 bis Node-3 erstellen:

```
gluster volume create glusterfs-1-volume transport tcp replica 3 192.168.1.61:/data/glusterfs 192.168.1.62:/data/glusterfs
192.168.1.63:/data/glusterfs force
```

Das GlusterFS Volume starten:

```
gluster volume start glusterfs-1-volume
```

Spezielle Einstellungen für größere VMs und große Images setzen aktivieren:

```
gluster volume set glusterfs-1-volume group virt
```

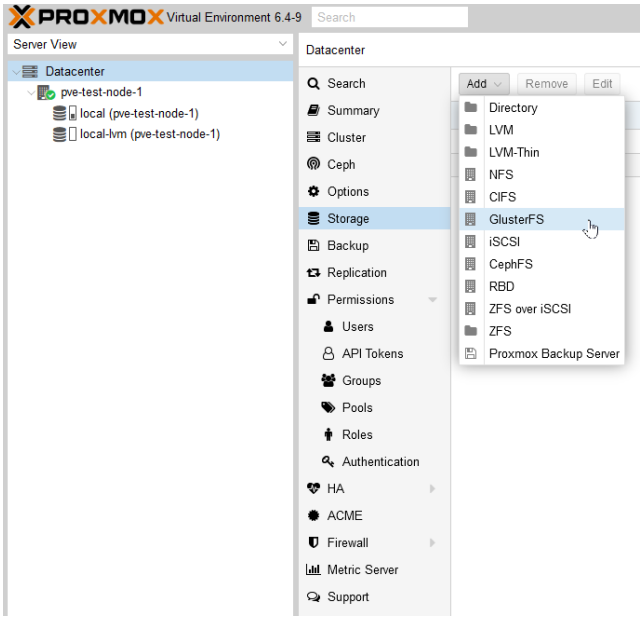
Diese danach überprüfen:

```
gluster volume info
```

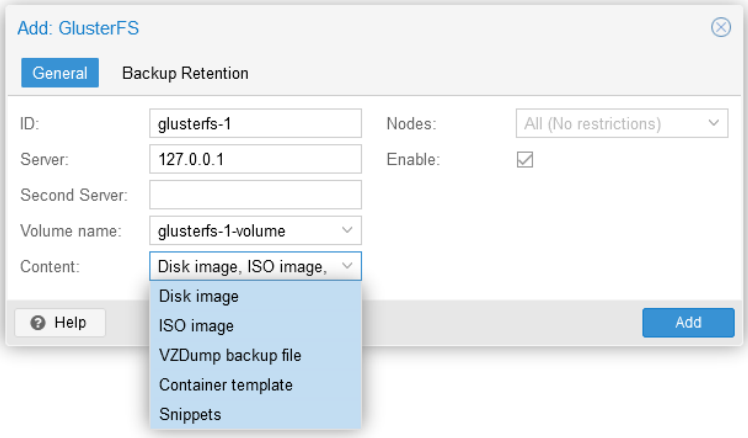
```
features.shard: on
```

TEIL 5 – GLUSTERFS-VOLUME IN PROXMOX HINZUFÜGEN

Über **Datacenter > Storage > GlusterFS** auf einem der Nodes einen neuen Storage des Typs GlusterFS erzeugen:

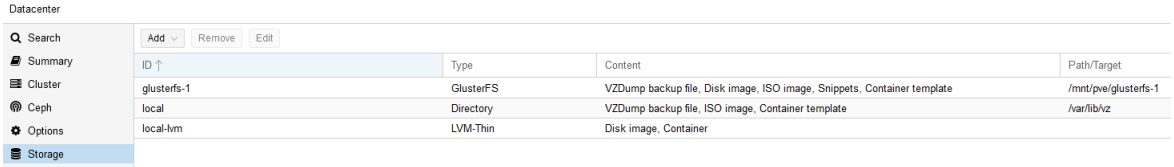


Den Storage mit den folgenden Daten anlegen. Als IP die des Localhost (= 127.0.0.1) nehmen. Somit verbindet sich jeder Proxmox-Node auf sich selbst und den Rest erledigt das GlusterFS-Cluster.



Dies wird, da Proxmox als Cluster konfiguriert ist, automatisch auf die anderen Nodes des Clusters synchronisiert.

Ein erfolgreicher angelegter GlusterFS Storage sieht in der WebUI dann so aus:



TEIL 6 – WORKAROUND: GLUSTERFS FÜR LXC BEREITSTELLEN

Aktuell können Images der Container (LXCs) im Format RAW nicht auf einem GlusterFS Storage abgelegt werden. Durch den folgenden Workaround wird dies möglich. Einen Nachteil gibt es allerdings: Es können keine Snapshots angelegt werden.

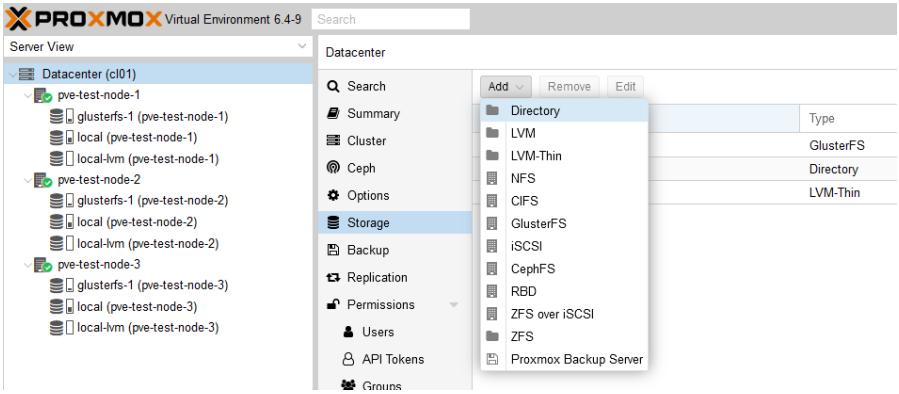
Die **/etc/fstab** um folgenden Eintrag, auf **jedem** der Nodes, erweitern und danach diesen in das System einhängen. Darauf achten, dass der Mount-Pfad (hier im Beispiel */mnt/pve/**glusterfs-1***) der ID / dem Namen entspricht, welcher vorherigen Kapitel „GlusterFS-Volume in Proxmox hinzufügen“ gewählt wurde:

```
echo "localhost:glusterfs-1-volume /mnt/pve/glusterfs-1 glusterfs defaults,_netdev 0 0" >> /etc/fstab
mount -a
```

Das System gibt dann die folgende Meldung aus, die ist allerdings nur eine Warnung:

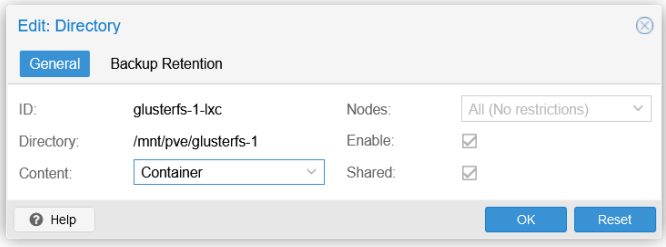
```
/sbin/mount.glusterfs: according to mtab, GlusterFS is already mounted on /mnt/pve/glusterfs-1
```

Über **Datacenter > Storage > Add** auf **einem** der Nodes einen neuen Storage vom Typ „Directory“ anlegen:



Die ID des neuen Directorys wird zum Namem des Storages. Unter Directory wird der in der /etc/fstab angebenen Pfad eingetragen.

Das Flag „Shared“ **muss manuell** aktiviert werden!



s

Beim Erstellen von LXCs kann nun **glusterfs-1-lxc** als Storage ausgewählt werden (siehe auch Kapitel „HA Konfiguration“).

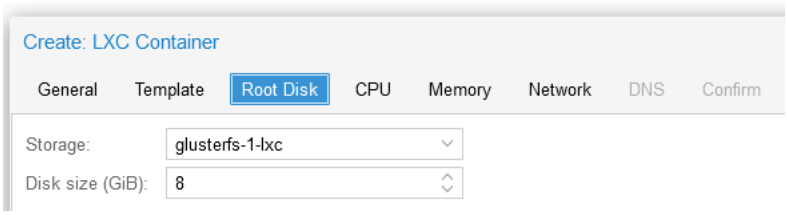
BEFEHLSÜBERSICHT

Befehl	Beschreibung
gstatus	Zeigt den Status des GlusterFS

gstatus -a -b	-a = all -b = list bricks
gluster volume status glusterfs-1-volume detail	
gluster volume status	Zeigt den Status der Volumes
gluster peer status	
gluster volume heal glusterfs-1-volume info	
gluster volume heal glusterfs-1-volume	
gluster volume heal glusterfs-1-volume info split-brain	

CONTAINER ERSTELLEN

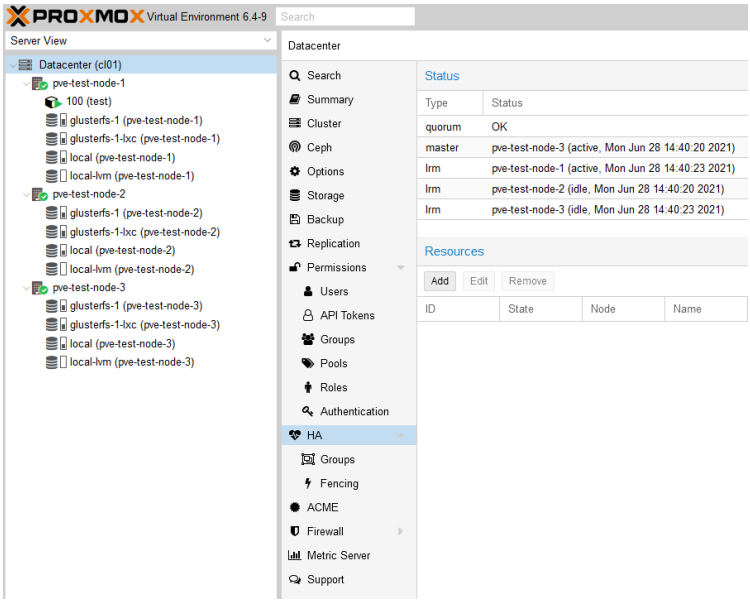
Wie im letzten Abschnitt des Kapitels „Container & VM“ erwähnt, werden nun die Container für das weitere Vorgehen erstellt. Beim Erstellen der Container darauf achten, dass diese auf dem GlusterFS-Storage erstellt, werden:



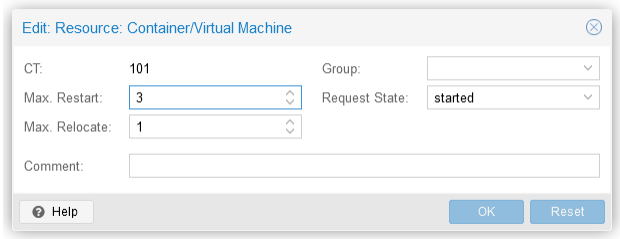
ID	Name	IP	OS
100	pve-test-iobroker	192.168.1.70	Debian 10
101	pve-test-redis-node-1	192.168.1.71	Debian 10
102	pve-test-redis-node-2	192.168.1.72	Debian 10
103	pve-test-redis-node-3	192.168.1.73	Debian 10

HA AKTIVIEREN

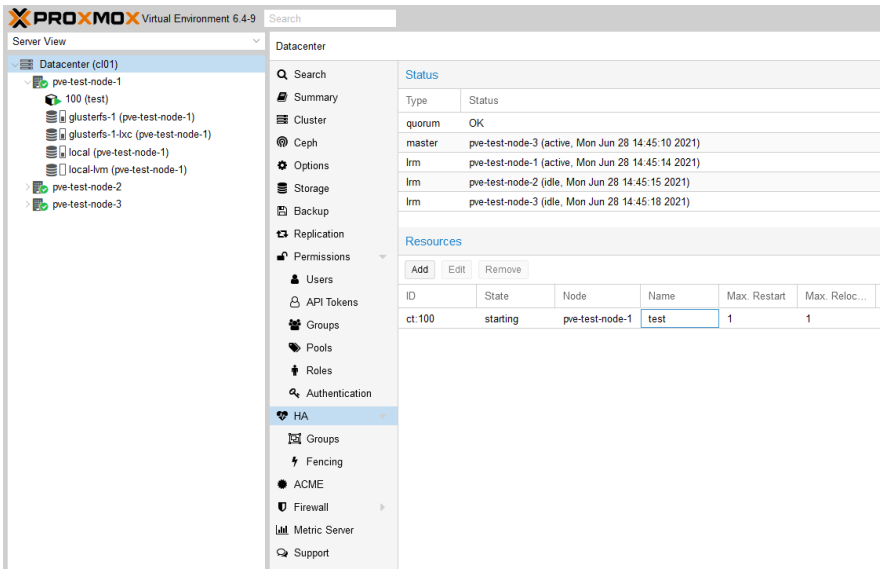
Auf einem der Proxmox-Nodes über **Datacenter > HA > Add** HA für die Container aktivieren:



Dazu unter **VM** die ID des LXCs oder der VM auswählen und die Max. Anzahl der Start-Versuche auf 3 setzen:

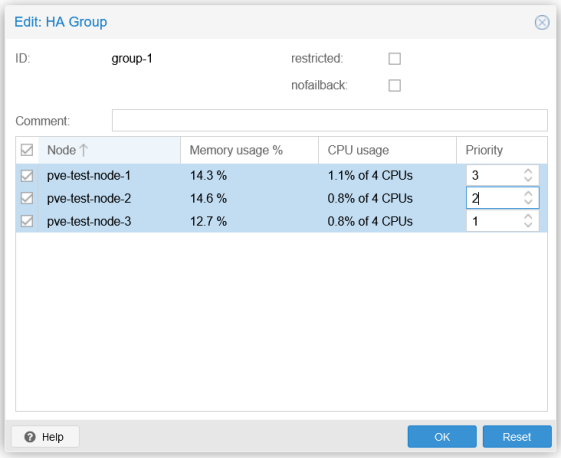


Danach ist HA für diesen Container aktiviert. Der LXC aus dem Beispiel liegt auf Node-1. Geht Node-1 offline, wird der LXC nach einer kurzen Zeit automatisch auf Node-2/-3 verschoben:



Über **Datacenter > HA > Groups** kann die Priorität für die einzelnen Nodes definiert werden. Eine größere Zahl entspricht einer höheren Priorität. Proxmox wird dann versuchen die VM/CT auf den Node mit der höchsten Priorität in einer Ausfallsituation zu migrieren.

Mehr Details dazu findet ihr hier: https://pve.proxmox.com/wiki/High_Availability#_configuration



Über das Flag **nofailback** kann gesteuert werden, ob auf den ausgefallenen Node, falls er wieder Verfügbar ist, der CT zurück migriert wird.

FEHLERBEHANDLUNG

HA: CT ODER VM HÄNGT IM STATUS „DELETING“

Auf allen Nodes den HA-Manager stoppen:

```
systemctl stop pve-ha-crm
```

Auf einem Node die HA-Konfiguration löschen:

```
rm /etc/pve/ha/manager_status
```

Auf allen Nodes den HA-Manager wieder starten:

```
systemctl start pve-ha-crm
```

https://forum.proxmox.com/threads/cannot-delete-ha-resources-since-pve-5-to-6-update.58151/#post-269561

FIREFOX: SEC_ERROR_REUSED_ISSUER_AND_SERIAL

Nach dem Beitreten zu einem Cluster, kann es beim Benutzen von Firefox dazu kommen, dass man die Web-Oberflächen von Node-2 und Node-3 nicht mehr erreichen kann. Dies liegt an ungültig gewordenen Zertifikaten, welche im Firefox Profil für diese Adressen hinterlegt worden sind. Diese können über

Einstellungen > Datenschutz & Sicherheit > Zertifikate > Zertifikate anzeigen > Server

entfernt werden. Danach Firefox beenden und im Ordner des Firefox-Profils die Dateien

```
cert9.db
cert_override.txt
```

löschen.

Danach Firefox neu Starten und der Zugriff sollte wieder funktionieren.

WEITERFÜHRENDES

GLUSTERFS RELEASE NOTES

In Teil 1 der Anleitung wird GlusterFS aktualisiert. Mit den Release Notes kann bei Bedarf die Version in diesem Schritt angepasst werden. Dazu die entsprechenden URLs aus den Release-Notes kopieren und entsprechend ersetzen. Aktuell setzt diese Anleitung auf die Version 8.x.

https://docs.gluster.org/en/latest/release-notes/

GLUSTERFS UPGRADE GUIDE V5 > V8

https://docs.gluster.org/en/latest/Upgrade-Guide/upgrade-to-8/

GLUSTERFS VOLUME OPTIONEN

https://access.redhat.com/documentation/en-us/red_hat_gluster_storage/3.1/html/administration_guide/chap-managing_red_hat_storage_volumes#Configuring_Volume_Options

PROXMOX ROADMAP

https://pve.proxmox.com/wiki/Roadmap

DETAILLIERTE BESCHREIBUNG CLUSTER ERSTELLEN

https://www.informaticar.net/how-to-setup-proxmox-cluster-ha/

ZUSÄTZLICHES BRICK HINZUFÜGEN

https://www.cyberciti.biz/faq/howto-add-new-brick-to-existing-glusterfs-replicated-volume

RASPBERRY PI ALS QUORUM DEVICE

https://www.youtube.com/watch?v=7-aZMyslkDg

PROXMOX-FORUM BEITRAG INSTALLATION GLUSTERFS

https://forum.proxmox.com/threads/create-a-proxmox-6-2-cluster-glusterfs-storage.71971/

PROXMOX-FORUM ARTIKEL LXC AUF GLUSTERFS STORAGE

https://forum.proxmox.com/threads/container-on-gluster-volume-not-possible.40889/

REDIS

VORRAUSETZUNGEN

3 Linux-Container (LXC) für die Redis-Server.

1. `pve-test-redis-node-1` (192.168.1.71)
2. `pve-test-redis-node-2` (192.168.1.72)
3. `pve-test-redis-node-3` (192.168.1.73)

Siehe auch Kapitel „Proxmox - Container und VMs“, letzter Abschnitt.

INSTALLATION & KONFIGURATION REDIS-SERVER

Am Ende dieses Abschnittes existieren 3 Redis Nodes welche sich untereinander replizieren.

AUF ALLEN NODES

```
apt install redis-server -y
systemctl enable redis-server
systemctl stop redis-server
```

Zuerst eine Sicherheitskopie der **redis.conf** anlegen:

```
cp /etc/redis/redis.conf /etc/redis/redis.conf.org
```

Optional: *Kommentare aus der **redis.conf** entfernen*:

```
grep -o '^[^#]*' /etc/redis/redis.conf > /etc/redis/redis.conf.no-comments
cp /etc/redis/redis.conf.no-comments /etc/redis/redis.conf
rm /etc/redis/redis.conf.no-comments
```

AUF PVE-TEST-REDIS-NODE-1

Die **redis.conf** wie folgt anpassen:

```
bind 0.0.0.0
protected-mode no
```

Alternativ, *Anpassen der **redis.conf** mit sed*:

```
sed -i "s|bind 127.0.0.1 ::1|bind 0.0.0.0|g" /etc/redis/redis.conf
sed -i "s|protected-mode yes|protected-mode no|g" /etc/redis/redis.conf
```

AUF PVE-TEST-REDIS-NODE-2/3

Die **redis.conf** wie folgt anpassen:

```
bind 0.0.0.0
protected-mode no
replicaof 192.168.1.71 6379
```

Alternativ, *Anpassen der **redis.conf** mit sed*:

```
sed -i "s|bind 127.0.0.1 ::1|bind 0.0.0.0|g" /etc/redis/redis.conf
sed -i "s|protected-mode yes|protected-mode no|g" /etc/redis/redis.conf
echo "replicaof 192.168.1.71 6379" >> /etc/redis/redis.conf
```

AUF ALLEN NODES

Den Dienst **redis-server** wieder starten:

```
systemctl start redis-server
```

ÜBERPRÜFEN DER INSTALLATION

```
redis-cli info replication
```

Ausgabe auf `pve-test-redis-node-1`:

```
# Replication
role:master
connected_slaves:2
slave0:ip=192.168.1.72,port=6379,state=online,offset=70,lag=1
slave1:ip=192.168.1.73,port=6379,state=online,offset=70,lag=1
master_replid:e53e73154cf03de4d8879eb2b4a7fdd9ea92bf98
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:70
second_repl_offset:-1
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:1
repl_backlog_histlen:7
```

Ausgabe auf `pve-test-redis-node-2` & `pve-test-redis-node-3`:

```
# Replication
role:slave
master_host:192.168.1.71
master_port:6379
master_link_status:up
master_last_io_seconds_ago:3
master_sync_in_progress:0
slave_repl_offset:196
slave_priority:100
slave_read_only:1
connected_slaves:0
master_replid:e53e73154cf03de4d8879eb2b4a7fdd9ea92bf98
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:196
second_repl_offset:-1
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:1
repl_backlog_histlen:196
```

TEST DER REPLIKATION

Auf dem aktuellen Master-Node, also initial der `pve-test-redis-node-1` einen neuen Datensatz erstellen:

```
redis-cli set db-test test123
```

Auf einem der anderen Nodes den Wert abfragen:

```
redis-cli get db-test
```

```
“test123”
```

Auf auf dem aktuellen Master (im Beispiel der `pve-test-redis-node-1`) kann dieser Datensatz mit

```
redis-cli del db-test
```

wieder gelöscht werden.

OPTIONAL: GETRENNTE REDIS PROZESSE FÜR FILE- UND OBJEKT-DB

"Am einfachsten ist es natürlich, States und Objekte zusammen in einem Redis-Prozess speichern zu lassen. Dies bedeutet allerdings auch dass nur alle Daten zusammen gesichert werden können. Bei der ioBroker File-DB waren States, Objekte und Files getrennt und konnten so selektiv gesichert werden. Auch die Schreiblast ist, wenn alles in einem Redis gespeichert ist, höher, da die Datenbank größer ist. – Apollon77 (https://forum.iobroker.net/topic/26327/redis-in-iobroker-%C3%BCberblick/2)

Dazu muss auf **jedem** der 3 Redis-Nodes, ein zusätzlicher Redis-Prozess konfiguriert werden.

Die folgenden Schritte sind dafür erforderlich:

Eine neue Ordner-Struktur für die Daten der zweiten Redis Instanz erstellen:

```
install -o redis -g redis -d /var/lib/redis2
```

Die bestehende **redis.conf** kopieren:

```
cp -p /etc/redis/redis.conf /etc/redis/redis2.conf
```

Die **redis2.conf** anpassen:

```
port 6380
pidfile /var/run/redis/redis-server2.pid
logfile /var/log/redis/redis-server2.log
dir /var/lib/redis2
```

Auf den Slave-Nodes in der **redis2.conf** auf die Zeile

```
replicaof 192.168.1.71 6380
```

achten!

*Alternativ, Anpassen der **redis2.conf** mit sed:*

```
sed -i "s|port 6379|port 6380|g" /etc/redis/redis2.conf
sed -i "s|pidfile /var/run/redis/redis-server.pid|pidfile /var/run/redis/redis-server2.pid|g" /etc/redis/redis2.conf
sed -i "s|logfile /var/log/redis/redis-server.log|logfile /var/log/redis/redis-server2.log|g" /etc/redis/redis2.conf
sed -i "s|dir /var/lib/redis|dir /var/lib/redis2|g" /etc/redis/redis2.conf
# nur auf den slave-nodes
sed -i "s|replicaof 192.168.1.71 6379|replicaof 192.168.1.71 6380|g" /etc/redis/redis2.conf
```

Das Systemd-Unit-File für die zweite Instanz erstellen, ...

```
cp /lib/systemd/system/redis-server.service /lib/systemd/system/redis-server2.service
```

... und anpassen:

```
ExecStart=/usr/bin/redis-server /etc/redis/redis2.conf
PIDFile=/run/redis/redis-server2.pid
ReadWriteDirectories=-/var/lib/redis2
Alias=redis2.service
```

Alternativ, wieder die Anpassung mit sed:

```
sed -i "s|ExecStart=/usr/bin/redis-server /etc/redis/redis.conf|ExecStart=/usr/bin/redis-server /etc/redis/redis2.conf|g"
/lib/systemd/system/redis-server2.service
sed -i "s|PIDFile=/run/redis/redis-server.pid|PIDFile=/run/redis/redis-server2.pid|g" /lib/systemd/system/redis-server2.service
sed -i "s|ReadWriteDirectories=-/var/lib/redis|ReadWriteDirectories=-/var/lib/redis2|g" /lib/systemd/system/redis-server2.service
sed -i "s|Alias=redis.service|Alias=redis2.service|g" /lib/systemd/system/redis-server2.service
```

Danach das Service-File aktivieren und das System durchstarten:

```
systemctl enable redis-server2.service
reboot
```

Mit

```
ps -ef | grep redis
```

kann überprüft werden, ob der zweite Prozess läuft:

```
redis      154     1  0 10:22 ?        00:00:00 /usr/bin/redis-server 0.0.0.0:6380
redis      155     1  0 10:22 ?        00:00:00 /usr/bin/redis-server 0.0.0.0:6379
```

Alternativ auch mit

```
systemctl status redis-server
systemctl status redis-server2
```

```
Active: active (running) since …
```

INSTALLATION & KONFIGURATION REDIS-SENTINEL

AUF ALLEN NODES

```
apt install redis-sentinel -y
systemctl enable redis-sentinel
systemctl stop redis-sentinel
```

Sicherheitskopie der **sentinel.conf** anlegen:

```
cp /etc/redis/sentinel.conf /etc/redis/sentinel.conf.org
```

Optional: *Kommentare aus der **sentinel.conf** entfernen*:

```
grep -o '^[^#]*' /etc/redis/sentinel.conf > /etc/redis/sentinel.conf.no-comments
cp /etc/redis/sentinel.conf.no-comments /etc/redis/sentinel.conf
rm /etc/redis/sentinel.conf.no-comments
```

Die **sentinel.conf** anpassen:

```
bind 0.0.0.0
sentinel failover-timeout mymaster 20000
sentinel monitor mymaster 192.168.1.71 6379 2
```

Optional, Anpassung mit sed und echo:

```
sed -i "s|bind 127.0.0.1 ::1|bind 0.0.0.0|g" /etc/redis/sentinel.conf
sed -i "s|sentinel monitor mymaster 127.0.0.1 6379 2|sentinel monitor mymaster 192.168.1.71 6379 2|g" /etc/redis/sentinel.conf
echo "sentinel failover-timeout mymaster 20000" >> /etc/redis/sentinel.conf
```

Beim Konfigurieren der **sentinel.conf** sicherstellen, dass die **myid** je Host unterschiedlich ist.

Der failover-timeout von 20 Sekunden (20000ms) sollte so gewählt sein, dass ein normaler Reboot die Failover-Situation nicht auslöst. Bei einem LXC dürften die 20 Sekunden gut passen.

Der Name des Masters lautet in der default Konfiguration: **mymaster**. Dieser wird ggf. bei Tests und Fehlersuche noch benötigt. Siehe auch Kapitel Befehlsübersicht.

Danach den Dienst **redis-sentinel** wieder starten:

```
systemctl start redis-sentinel
```

ÜBERPRÜFEN DER INSTALLATION

```
systemctl status redis-sentinel

redis-cli -p 26379 info sentinel
```

Ausgabe auf `pve-test-redis-node-1`:

```
# Sentinel
sentinel_masters:1
sentinel_tilt:0
sentinel_running_scripts:0
sentinel_scripts_queue_length:0
sentinel_simulate_failure_flags:0
master0:name=mymaster,status=ok,address=192.168.1.71:6379,slaves=2,sentinels=3
```

Die Anzahl der Sentinels in der letzten Zeile muss der Anzahl der redis-sentinel Instanzen entsprechen – im Beispiel hier, 3 Stück.

FAILOVER-TEST

Auf `pve-test-redis-node-2` oder `pve-test-redis-node-3` das redis-sentinel.log aufrufen:

```
tail -f /var/log/redis/redis-sentinel.log
```

Dann den aktuellen Redis-Master LXC (`pve-test-redis-node-1`) stoppen und das Log beobachten. Darin sollte nach rund 20 Sekunden (siehe sentinel.conf failover-timeout) eine Neuwahl eines Masters zu beobachten sein. Alternativ zum Stoppen des LXCs kann auch mittels `redis-cli debug sleep 60` der Redis-Prozess pausiert werden.

```
166:X 13 Jul 2021 19:45:16.005 # +sdown master mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.005 # +sdown sentinel 98ebf650b75f69796b54610aa4fca802d86d555e 192.168.1.71 26379 @ mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.076 # +odown master mymaster 192.168.1.71 6379 #quorum 2/2
166:X 13 Jul 2021 19:45:16.077 # +new-epoch 1
166:X 13 Jul 2021 19:45:16.077 # +try-failover master mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.112 # +vote-for-leader 949af74096102af17e79ca75fc0294d78eb0bdae 1
166:X 13 Jul 2021 19:45:16.184 # 09a2fa6d613acfa3527de0c3ab0fbc2ef1efa2e1 voted for 949af74096102af17e79ca75fc0294d78eb0bdae 1
166:X 13 Jul 2021 19:45:16.189 # +elected-leader master mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.189 # +failover-state-select-slave master mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.260 # +selected-slave slave 192.168.1.72:6379 192.168.1.72 6379 @ mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.260 * +failover-state-send-slaveof-noone slave 192.168.1.72:6379 192.168.1.72 6379 @ mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.316 * +failover-state-wait-promotion slave 192.168.1.72:6379 192.168.1.72 6379 @ mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.482 # +promoted-slave slave 192.168.1.72:6379 192.168.1.72 6379 @ mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.482 # +failover-state-reconf-slaves master mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.555 * +slave-reconf-sent slave 192.168.1.73:6379 192.168.1.73 6379 @ mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.846 * +slave-reconf-inprog slave 192.168.1.73:6379 192.168.1.73 6379 @ mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.846 * +slave-reconf-done slave 192.168.1.73:6379 192.168.1.73 6379 @ mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.921 # +failover-end master mymaster 192.168.1.71 6379
166:X 13 Jul 2021 19:45:16.921 # +switch-master mymaster 192.168.1.71 6379 192.168.1.72 6379
166:X 13 Jul 2021 19:45:16.921 * +slave slave 192.168.1.73:6379 192.168.1.73 6379 @ mymaster 192.168.1.72 6379
```

```
166:X 13 Jul 2021 19:45:16.921 * +slave slave 192.168.1.71:6379 192.168.1.71 6379 @ mymaster 192.168.1.72 6379
166:X 13 Jul 2021 19:45:46.997 # +sdown slave 192.168.1.71:6379 192.168.1.71 6379 @ mymaster 192.168.1.72 6379
```

OPTIONAL: SENTINEL FÜR MEHRERE REDIS-PROZESSE KONFIGURIEREN

Wenn im vorherigen Kapitel mehrere Redis-Prozesse konfiguriert wurden, muss auch der Redis-Sentinel entsprechend angepasst werden. Allerdings braucht es dazu keinen eigenen redis-sentinel Prozess. Es reicht aus, die Konfiguration in der **sentinel.conf, auf jedem Node** zu verdoppeln.

Zuerst Redis-Sentinel Prozess stoppen:

```
systemctl stop redis-sentinel
```

Die Konfiguration anpassen:

```
sentinel monitor objects-master 192.168.1.71 6379 2
sentinel failover-timeout objects-master 20000
sentinel config-epoch objects-master 0
sentinel leader-epoch objects-master 0

sentinel monitor states-master 192.168.1.71 6380 2
sentinel failover-timeout states-master 20000
sentinel config-epoch states-master 0
sentinel leader-epoch states-master 0
```

Die **sentinel.conf** sollte dann so aussehen:

Achtung bei Copy+Paste: Die myid muss eindeutig sein und sollte nicht überschrieben werden!

```
bind 0.0.0.0
port 26379
daemonize yes
pidfile "/var/run/sentinel/redis-sentinel.pid"
logfile "/var/log/redis/redis-sentinel.log"
dir "/var/lib/redis"
protected-mode no

sentinel myid 9ff532162c4b9f660839aae047c827533b8b1f59
sentinel deny-scripts-reconfig yes

sentinel monitor objects-master 192.168.1.71 6379 2
sentinel failover-timeout objects-master 20000
sentinel config-epoch objects-master 0
sentinel leader-epoch objects-master 0

sentinel monitor states-master 192.168.1.71 6380 2
sentinel failover-timeout states-master 20000
sentinel config-epoch states-master 0
sentinel leader-epoch states-master 0
```

Danach den Redis-Sentinel Prozess wieder starten:

```
systemctl start redis-sentinel
```

Nach dem Start von redis-sentinel wird die sentinel.conf mit weiteren Daten des Sentinels-Cluster erweitert:

```
# Generated by CONFIG REWRITE
…
sentinel known-replica objects-master 192.168.1.72 6379
sentinel known-replica objects-master 192.168.1.73 6379
sentinel current-epoch 0
…
```

Die Redis Instanz auf Port 6379 stellt somit die Datenbank für die ioBroker objects bereit, die Redis Instanz auf Port 6380 die Datenbank für die ioBroker states. Die Konfiguration des ioBrokers erfolgt im nächsten Kapitel.

Überprüfen, ob der Sentinel läuft:

```
systemctl status redis-sentinel
```

```
Active: active (running)
```

```
redis-cli -p 26379 info sentinel
```

```
# Sentinel
sentinel_masters:2
sentinel_tilt:0
sentinel_running_scripts:0
sentinel_scripts_queue_length:0
sentinel_simulate_failure_flags:0
master0:name=states-master,status=ok,address=192.168.1.71:6380,slaves=2,sentinels=3
master1:name=objects-master,status=ok,address=192.168.1.71:6379,slaves=2,sentinels=3
```

BEFEHLSÜBERSICHT

Im Standard benutzt redis-cli den Port 6379. Sind, wie optional beschrieben, mehrere Redis-Instanzen je Container aufgesetzt, kann auf die zweite Instanz über den Port 6380 zugegriffen werden. Über den Port 26379 wird der redis-sentinel abgefragt.

Befehl	Beschreibung
redis-cli info replication	Informationen des aktuellen Zustandes der Replikation
redis-cli -p 6379 debug sleep 60	Pausiert den redis-Prozess
redis-cli -p 26379 info sentinel	Informationen zum redis-sentinel
redis-cli -p 26379 sentinel get-master-addr-by-name <master-name>	Zeigt den redis-master von <master-name> Anstelle von <master-name>: objects-master / states-master bei getrennten Redis-Prozessen
redis-cli -p 26379 sentinel sentinels mymaster	
redis-cli -p 26379 sentinel slaves mymaster	

WEITERFÜHRENDES

Redis Dokumentation

https://redis.io/topics/quickstart

Redis Sentinel Dokumentation

https://redis.io/topics/sentinel

Überblick Redis und ioBroker

https://forum.iobroker.net/topic/26327/redis-in-iobroker-%C3%BCberblick

Redis Replica / HA Howto (Achtung: Das weicht zum Teil von diesem hier ab!)

https://www.tecmint.com/setup-redis-replication-in-centos-8
https://www.tecmint.com/setup-redis-high-availability-with-sentinel-in-centos-8

IOBROKER

Den ioBroker mit

```
curl -sL https://iobroker.net/install.sh | sudo bash
```

installieren. Damit werden Node JS / npm und ioBroker installiert.

Danach den ioBroker wieder stoppen und Konfigurieren:

```
iobroker stop
iobroker setup custom
```

```
Current configuration:
- Objects database:
  - Type: file
  - Host/Unix Socket: 127.0.0.1
  - Port: 9001
- States database:
  - Type: file
  - Host/Unix Socket: 127.0.0.1
  - Port: 9000
- Data Directory: ../../iobroker-data/
```

Wichtig: Es muss der Port des Sentinels angegeben werden: 26379

Als Host werden die IPs der Redis-Nodes angegeben: 192.168.1.71,192.168.1.72,192.168.1.73 – somit erkennt der ioBroker auch das es sich um eine Sentinel-Konfiguration handelt und frägt den Namen der DBs ab.

```
Type of objects DB [(f)ile, (r)edis, ...], default [file]: r

When Objects and Files are stored in a Redis database please consider the following:

1. All data will be stored in RAM, make sure to have enough free RAM available!
2. Make sure to check Redis persistence options to make sure a Redis problem will not cause data loss!
3. The Redis persistence files can get big, make sure not to use an SD card to store them.

Host / Unix Socket of objects DB(redis), default[127.0.0.1]: 192.168.1.71,192.168.1.72,192.168.1.73
Port of objects DB(redis), default[26379]: 26379
Objects Redis Sentinel Master Name [mymaster]: objects-master
Type of states DB [(f)file, (r)edis, ...], default [redis]: r
Host / Unix Socket of states DB (redis), default[192.168.1.71,192.168.1.72,192.168.1.73]: <Enter>
Port of states DB (redis), default[26379]: <Enter>
States Redis Sentinel Master Name [objects-master]: states-master
Host name of this machine [pve-test-iobroker]: <Enter>
Please choose if this is a Master/single host (enter "m") or a Slave host (enter "S") you are about to edit. For Slave hosts the data
migration will be skipped. [S/m]: m

Important: Using redis for the Objects database is only supported with js-controller 2.0 or higher!

When your system consists of multiple hosts please make sure to have js-controller 2.0 or higher installed on ALL hosts *before*
continuing!

Important #2: If you already did the migration on an other host please *do not* migrate again! This can destroy your system!
```

```
Important #3: The process will migrate all files that were officially uploaded into the ioBroker system. If you have manually copied
files into iobroker-data/files/... into own directories then these files will NOT be migrated! Make sure all files are in adapter
directories inside the files directory!

Do you want to migrate objects and states from "file/file" to "redis/redis" [y/N]: y

Migrating the objects database will overwrite all objects! Are you sure that this is not a slave host and you want to migrate the
data? [y/N]: y
```

Danach den ioBroker wieder starten:

```
iobroker start
```

FERTIG! 😊