DOKUMENTATION: FAILOVER-CLUSTER MIT DEBIAN 9.2

Projekt:	INTERN-TEST	Erster Stand:	18.10.2017
Herausgeber:	Hellberg EDV Beratung	Letzter Stand:	07.12.2017
Alle Bearbeiter:	MT, TS, PP,GH	Letzter Bearbeiter:	GH
Dokumentation:			
Kurzbeschreibung:	Failover-Cluster mit 2 Knoten mit Debian 9.2, DRBD, Corosync, Pacemaker und NFS		

AUSGANGSLAGE / ZIELSETZUNG:

Diese Dokumentation 2-Knoten Failover-Cluster mit Debian 9.2 wurde überarbeitet, nachdem die 3-Knoten-Failover-Cluster-Version bereits erstellt und getestet wurde. Mit dem neuen Kenntnisstand wurde diese Dokumentation entsprechend überarbeitet.

Das Ziel dieser Dokumentation ist vorerst ein Failover-Cluster mit 2 Knoten für eine NFS-Freigabe mit 2 virtuellen Maschinen. Die Blöcke einer virtuellen Festplatte sollen per DRBD von einem System auf das andere gespiegelt werden. Für die Kommunikation zwischen den 2 Knoten soll Corosync und für die Verwaltung der Ressourcen (z. B. Cluster IP, Festlegung primary/secondary) Pacemaker verwendet werden. Per NFS soll auf der gespiegelten virtuellen Festplatte eine Netzwerkfreigabe erstellt werden, die bei Ausfall des primären Knotens automatisch auf den zweiten Knoten übertragen wird. Weiter soll der Failover-Cluster eine virtuelle IP zugewiesen bekommen, damit dieser für Clients immer unter derselben Adresse erreichbar ist.

Folgende Dinge können noch behandelt werden:

- Verschlüsselung Corosync (z.B mit Wireshark prüfen)

HINWEISE

Um die Knoten ordnungsgemäß herunterfahren zu können, muss vorher der Dienst Pacemaker gestoppt werden.

Anmerkung: Failover findet auch in diesem Fall statt.

```
service pacemaker stop
```

1. Knoten:

Hostname:	ha1			
Domain:	sesam.de			
IP:	4.4.4.10/24			
DNS:	-			
GW:	-			
Eintragungen hosts:	4.4.4.10	ha1.sesam.de	ha1	
	4.4.4.20	ha2.sesam.de	ha2	
2 vHDDs:	/dev/sda		100GB	
	/dev/sdb		15GB	
Partitionierung:	/dev/sda1	/boot	1GB	ext4
	/dev/sda2	swap	10GB	
	/dev/sda3	/	50GB	ext4
	/dev/sda4	/daten	41GB	ext4
	/dev/sdb1		15GB	
	-/dev/drbd0			ext4
Installierte Dienste:	DRBD		8.9.10-2	
	Corosync		2.4.2-3	
	Pacemaker		1.1.16-1	
	crmsh		2.3.2-4	
	NFS-kernel-server		1.3.4-2.1	

2. Knoten:

Hostname:	ha2			
Domain:	sesam.de			
IP:	4.4.4.20/24			
DNS:	-			
GW:	-			
Eintragungen hosts:	4.4.4.20	ha2.sesam.de	ha2	
	4.4.4.10	ha1.sesam.de	ha1	
2 vHDDs:	/dev/sda	100GB		
	/dev/sdb	15GB		
Partitionierung:	/dev/sda1	/boot	1GB	ext4
	/dev/sda2	swap	10GB	
	/dev/sda3	/	50GB	ext4
	/dev/sda4	/daten	41GB	ext4
	/dev/sdb1		15GB	
	-/dev/drbd0			ext4
Installierte Dienste:	DRBD	8.9.10-2		
	Corosync	2.4.2-3		
	Pacemaker	1.1.16-1		
	crmsh	2.3.2-4		
	NFS-kernel-server	1.3.4-2.1		

SCHAUBILD:

Siehe Vorlesungen.

STEP 1: DIENSTE DRBD, COROSYNC, PACEMAKER UND NFS AUS AUTOSTART ENTFERNEN

Die Dienste DRBD, Corosync und Pacemaker werden aus dem Autostart entfernt, da sie zuerst Konfiguriert werden müssen. NFS wird ebenfalls aus dem Autostart entfernt, da dieser später von Pacemaker gestartet wird.

```
systemctl disable <Dienstname>
```

Die Befehle werden wie folgt eingegeben: (rote Kästchen)

```
hedv@hal:~$ su
Passwort:
root@hal:/home/hedv# systemctl disable pacemaker
Synchronizing state of pacemaker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable pacemaker
root@hal:/home/hedv# systemctl disable corosync
Synchronizing state of corosync.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable corosync
root@hal:/home/hedv# systemctl disable drbd
drbd.service is not a native service, redirecting to systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable drbd
root@hal:/home/hedv# systemctl disable nfs-kernel-server
Synchronizing state of nfs-kernel-server.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable nfs-kernel-server
Removed /etc/systemd/system/multi-user.target.wants/nfs-server.service.
root@hal:/home/hedv# █
```

Um zu überprüfen, ob ein Dienst im Autostart ist, wird folgender Befehl ausgeführt.

```
systemctl is-enabled <Dienstname>; echo $?
```

0 = ist im Autostart (enabled)

1= ist nicht im Autostart (disabled)

Anschließend werden die Systeme neugestartet.

STEP 2: DRBD-KONFIGURATION

Konfiguration in Datei `/etc/drbd.d/r0.res` ist auf beiden Knoten (ha1 und ha2) identisch.

```
resource r0 {
    on ha1 {
        device      /dev/drbd0;
        disk        /dev/sdb1;
        address     4.4.4.10:7789;
        meta-disk   internal;
    }
    on ha2 {
        device      /dev/drbd0;
        disk        /dev/sdb1;
        address     4.4.4.20:7789;
        meta-disk   internal;
    }
}
```

Jetzt wird die Ressource durch folgende Befehle erstellt:

Auf ha1 und ha2:

```
drbdadm create-md r0

drbdadm up r0
```

Auf ha1:

```
drbdadm primary --force r0
```

Nach Ausführung der Befehle startete die Synchronisation von `/dev/drbd0` von ha1 zu ha2.

```
Every 2,0s: cat /proc/drbd                                     ha1: Wed Oct 18 13:44:30 2017
version: 8.4.7 (api:1/proto:86-101)
srcversion: 0904DF2CCF7283ACE07D07A
0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----
   ns:15727100 nr:0 dw:0 dr:15727100 al:8 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```

Und **/dev/drbd0** ist auf **/dev/sdb1** aufgesetzt:

```
root@hal:/etc/drbd.d# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
fd0         2:0    1    4K  0 disk
sda         8:0    0  100G  0 disk
├─sda1      8:1    0   953M  0 part /boot
├─sda2      8:2    0    9,3G  0 part [SWAP]
├─sda3      8:3    0   46,6G  0 part /
├─sda4      8:4    0   43,2G  0 part /daten
sdb         8:16   0    15G  0 disk
├─sdb1      8:17   0    15G  0 part
│   └─drbd0 147:0   0    15G  0 disk
sr0        11:0    1    3,7G  0 rom  /media/cdrom0
root@hal:/etc/drbd.d# █
```

STEP 3: REPLIZIERTES LAUFWERK MOUNTEN

Zuerst wird das logische Gerät **/dev/drbd0** mit dem Dateisystem **ext4** formatiert

```
mkfs.ext4 /dev/drbd0
```

```
root@hal:/dev# mkfs.ext4 /dev/drbd0
mke2fs 1.43.4 (31-Jan-2017)
Ein Dateisystems mit 3931775 (4k) Blöcken und 983040 Inodes wird erzeugt.
UUID des Dateisystems: 78ff5073-17d5-418e-a748-3cbdc930f08
Superblock-Sicherungskopien gespeichert in den Blöcken:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

beim Anfordern von Speicher für die Gruppentabellen: erledigt
Inode-Tabellen werden geschrieben: erledigt
Das Journal (16384 Blöcke) wird angelegt: erledigt
Die Superblöcke und die Informationen über die Dateisystemnutzung werden
geschrieben: erledigt
root@hal:/dev#
```

```
mount /dev/drbd0 /cluster
```

und anschließend in das manuell angelegte Verzeichnis **/cluster** eingehängt.

```
root@hal:/# mount /dev/drbd0 /cluster
root@hal:/# df -h
Dateisystem    Größe Benutzt Verf. Verw% Eingehängt auf
udev           2,0G      0  2,0G   0% /dev
tmpfs          396M    6,4M  390M   2% /run
/dev/sda3      46G     3,8G   40G   9% /
tmpfs          2,0G    54M   1,9G   3% /dev/shm
tmpfs          5,0M    4,0K   5,0M   1% /run/lock
tmpfs          2,0G      0  2,0G   0% /sys/fs/cgroup
/dev/sda4      43G    53M   41G   1% /daten
/dev/sda1     922M    36M  823M   5% /boot
tmpfs          396M    28K  396M   1% /run/user/115
tmpfs          396M    16K  396M   1% /run/user/1000
/dev/sr0        3,8G    3,8G   0 100% /media/cdrom0
/dev/drbd0     15G    41M   14G   1% /cluster
root@hal:/# █
```

STEP 4: KONFIGURATION COROSYNC

Konfiguration in Datei **/etc/corosync/corosync.conf** ist auf beiden Knoten (ha1 und ha2) identisch.

```

totem {
    version: 2
    token: 3000
    token_retransmits_before_loss_const: 10
    clear_node_high_bit: yes
    crypto_cipher: none
    crypto_hash: none
    transport: udpu
    interface {
        ringnumber: 0
        bindnetaddr: 4.4.4.0
    }
}

logging {
    to_logfile: yes
    logfile: /var/log/corosync/corosync.log
    debug: on
    timestamp: on
    logger_subsys {
        subsys: QUORUM
        debug: on
    }
}

quorum {
    provider: corosync_votequorum
    two_node: 1
    wait_for_all: 1
}

nodelist {
    node{
        ring0_addr: 4.4.4.10
    }
    node{
        ring0_addr: 4.4.4.20
    }
}

```

Damit die Kommunikation zwischen den Knoten von Corosync verschlüsselt stattfindet, wird auf ha1 zum Anlegen eines Corosync-Authentifizierungsschlüssels folgender Befehl ausgeführt:

```
corosync-keygen -l
```

Dieser Befehl erzeugt die Datei **/etc/corosync/authkey** und muss auf ha2 kopiert werden. Wir haben dafür scp benutzt, allerdings muss dafür der root-login per ssh auf ha2 aktiviert werden, da bei Debian 9 dieser standardmäßig deaktiviert ist. Dazu wird in der Datei **/etc/ssh/sshd_config** folgende Zeile angepasst:

```
#PermitRootLogin prohibit-password          zu          PermitRootLogin yes
```

Anschließend muss der Dienst ssh neugestartet werden. Danach wird der Corosync-authkey von ha1 auf ha2 kopiert:

```
scp /etc/corosync/authkey root@ha2:/etc/corosync/authkey
```

STEP 5: NFS-KERNEL-SERVER KONFIGURATION

Für diesen Schritt wird ha1 als primary für die DRBD-Ressource gesetzt, da die Konfigurationsdateien von NFS auf /dev/drbd0 kopiert werden müssen. **/dev/drbd0** wird auf **/cluster** eingehängt.

Im Verzeichnis **/cluster** werden folgende Verzeichnisse einzeln per mkdir erstellt:

```
var/lib/nfs/  
etc/
```

Als nächstes werden auf dem System ein Verzeichnis und eine Datei zwecks Beibehalten der Originale umbenannt.

```
mv /var/lib/nfs /var/lib/nfs-original  
mv /etc/exports /etc/exports-original
```

Dann werden die Konfigurationsdateien auf /dev/drbd0 kopiert.

```
cp -r --preserve=all /var/lib/nfs-original/* /cluster/var/lib/nfs  
cp --preserve=all /etc/exports-original /cluster/etc/exports
```

Weiter werden auf das kopierte Verzeichnis und die kopierte Datei symbolische Links erzeugt:

```
ln -s /cluster/var/lib/nfs /var/lib/nfs  
ln -s /cluster/etc/exports /etc/exports
```

```
lrwxrwxrwx 1 root root 20 Okt 20 12:12 exports -> /cluster/etc/exports
```

```
lrwxrwxrwx 1 root root 20 Okt 20 12:12 nfs -> /cluster/var/lib/nfs
```

Im Verzeichnis **/cluster** wird ein Verzeichnis für die NFS-Freigabe erstellt und für Testzwecke mit den Berechtigungen 777 versehen

```
mkdir nfsshare  
chmod -R 777 nfsshare/
```

und danach in der Datei **/cluster/etc/exports** die NFS-Freigabe erstellt.

```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes     hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4      gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
```

```
/cluster/nfsshare 4.4.4.0/24(rw,sync)|
```

Dann wird **/dev/drbd0** auf ha1 aufgehängt und für die DRBD-Ressource r0 auf secondary gesetzt

```
umount /dev/drbd0
drbdadm secondary r0
```

und ha2 dann als primary für die Ressource gesetzt und **/dev/drbd0** dort eingehängt.

```
drbdadm primary r0
mount /dev/drbd0 /cluster
```

Auf ha2 werden ebenfalls das Verzeichnis **/var/lib/nfs** und die Datei **/etc/exports** umbenannt

```
mv /var/lib/nfs /var/lib/nfs-original
mv /etc/exports /etc/exports-original
```

und ebenfalls die symbolischen Links, wie auf ha1, gesetzt.

```
ln -s /cluster/var/lib/nfs /var/lib/nfs
ln -s /cluster/etc/exports /etc/exports
```

STEP 6: KONFIGURATION PACEMAKER

Hierfür werden auf ha1 und ha2 die Dienste Corosync und Pacemaker gestartet.

```
service corosync start
service pacemaker start
```

Pacemaker wird über die crm-shell konfiguriert, diese wird mit dem Befehl **crm** gestartet. In der crm-shell „**configure**“ eingeben, um in den Konfigurationsmodus zu gelangen. In diesem werden folgende Befehle Zeile für Zeile eingegeben (ein \ + Enter gilt in der crm-shell als Weiterführung der Zeile)

Als erstes wird stonith (shoot the other node in the head) deaktiviert. Beim Stonith handelt es sich um ein Fencing-Mechanismus, der es Pacemaker erlaubt, den anderen Knoten neuzustarten wenn dieser Probleme macht. Mit no-quorum-policy=freeze werden die Ressourcen beim Starten des Clusters solange nicht gestartet, bis der zweite Knoten hochgefahren ist und Pacemaker den Cluster für funktional hält. Erst dann werden die Ressourcen gestartet.

```
property stonith-enabled=false
property no-quorum-policy=freeze
```

Die Ressource mit dem Namen „cluster_ip“ legt eine virtuelle IP fest, über die der Cluster später erreichbar sein wird.

```
primitive cluster_ip ocf:heartbeat:IPaddr \
op monitor interval="10" timeout="20" \
params ip="4.4.4.30" cidr_netmask="24"
```

Die Ressource „drbd_res“ verwaltet die DRBD-Ressource „r0“

```
primitive drbd_res ocf:linbit:drbd params drbd_resource="r0" \
op monitor interval="29s" role="Master" op monitor interval="31s" role="Slave"
```

Die Ressource „ms_drbd“ legt fest, dass es in der Ressource „drbd_res“ nur einen Master Knoten (primary) geben darf. Der Parameter „meta“ bedeutet, dass die 2 Knoten sich innerhalb des DRBD gegenseitig über etwaige Statusänderungen informieren.

```
ms ms_drbd drbd_res meta master-node-max="1" clone-max="2" clone-node-max="1" \
globally-unique="false" notify="true" target-role="Master"
```

Die Ressource „filesystem_res“ kümmert sich um das automatische Einhängen des Gerätes /dev/drbd0 in dem Verzeichnis /cluster auf dem primary.

```
primitive filesystem_res ocf:heartbeat:Filesystem \
op monitor interval="120" \
params device="/dev/drbd0" fstype="ext4" \
directory="/cluster"
```

Die Ressource „nfs_server_res“ startet den nfs-kernel-server-Dienst.

```
primitive nfs_server_res ocf:heartbeat:nfsserver \  
op monitor interval=30 \  
meta target-role=Started \  
params nfs_ip=4.4.4.30 nfs_shared_infodir=/cluster/var/lib/nfs
```

Die Ressource „fs_nfs_ip_grp“ fasst die einzelnen Ressourcen „filesystem_res“, „nfs_server_res“ und „cluster_ip“ zu einer Gruppe zusammen und legt die Startreihenfolge der Ressourcen fest.

```
group fs_nfs_ip_grp filesystem_res nfs_server_res cluster_ip
```

Die Ressource „col_fs_nfs_ip_grp_on_master“ legt fest, dass die Ressourcen innerhalb der Gruppe nur auf dem primary aktiv sein dürfen.

```
colocation col_fs_nfs_ip_grp_on_master inf: fs_nfs_ip_grp:Started ms_drbd:Master
```

Die Ressource „ord_fs_nfs_ip_grp_after_drbd_master“ legt fest, dass die Ressourcen in der Gruppe „fs_nfs_ip_grp“ auf einem Knoten erst gestartet werden würden, nachdem dieser zum primary befördert wurde.

```
order ord_fs_nfs_ip_grp_after_drbd_master inf: ms_drbd:promote fs_nfs_ip_grp:start
```

Die Ressource Location sorgt dafür, dass ha1, sobald dieser verfügbar ist, automatisch Primary wird. (Auch nach einem Ausfall)

```
location pref_ha1 ms_drbd rule role="master" 100: #uname eq ha1
```

Zuletzt werden die Konfigurationen per „commit“ bestätigt und die crm-shell mit „bye“ verlassen.

Der Befehl „crm status“ sollte danach folgende Ausgabe zeigen:

Anmerkung: Screenshot ist nicht aktuell, dank der location Ressource wird der Knoten ha1 automatisch zum Master, sobald dieser verfügbar ist.

```
root@ha1:/home/hedv# crm status
Stack: corosync
Current DC: ha2 (version 1.1.16-94ff4df) - partition with quorum
Last updated: Fri Oct 20 14:45:57 2017
Last change: Fri Oct 20 14:37:11 2017 by root via cibadmin on ha1
```

```
2 nodes configured
5 resources configured
```

```
Online: [ ha1 ha2 ]
```

```
Full list of resources:
```

```
Master/Slave Set: ms_drbd [drbd_res]
```

```
Masters: [ ha2 ]
```

```
Slaves: [ ha1 ]
```

```
Resource Group: fs_nfs_ip_grp
```

```
filesys_res (ocf::heartbeat:Filesystem): Started ha2
```

```
nfs_server_res (lsb:nfs-kernel-server): Started ha2
```

```
cluster_ip (ocf::heartbeat:IPaddr): Started ha2
```

```
root@ha1:/home/hedv# █
```

STEP 7: AUTOSTARTREIHENFOLGE DER DIENSTE DRBD, COROSYNC UND PACEMAKER

Da DRBD nicht nativ mit systemd kompatibel ist, funktioniert die von systemd automatisch konfigurierte Startreihenfolge der Dienste nicht. Die Dienste müssen in der Reihenfolge DRBD → Corosync → Pacemaker gestartet werden, systemd startet im Ausgangszustand aber Corosync und Pacemaker vor DRBD.

Dafür wird eine Konfigurationsdatei für systemd benötigt. In dieser wird konfiguriert, nach welchen Diensten DRBD gestartet werden soll und welche Befehle beim Starten oder Stoppen des Dienstes ausgeführt werden sollen.

```
[Unit]
Description=Distributed Replicated Block Device
After=systemd-modules-load.service network.target network-online.target

[Service]
Type=oneshot
RemainAfterExit=yes

# Check the configuration file for syntax errors
ExecStartPre=/sbin/drbdadm sh-nop

# Create the resources (and log what is being done)
ExecStart=/sbin/drbdadm adjust-with-progress all

# Wait for connection
ExecStart=/sbin/drbdadm wait-connect all

# Switch resources to primary if applicable
ExecStart=/sbin/drbdadm sh-b-pri all

# Deactivate all resources on stop
ExecStop=/sbin/drbdadm down all

[Install]
WantedBy=multi-user.target
```

Auf ha1 und ha2 wurde im Verzeichnis **/lib/systemd/system/** diese Konfigurationsdatei „**drbd.service**“ erstellt/kopiert.

Zusätzlich muss ein Symbolischer Link von **/lib/systemd/system/drbd.service** auf **/lib/systemd/system/multi-user.target.wants/drbd.service** erstellt werden.

Abschließend werden alle Dienste wieder in den Autostart gepackt.

```
systemctl enable drbd

systemctl enable corosync

systemctl enable pacemaker
```

Nach einem Neustart der zwei Knoten werden die Dienste dann in der korrekten Reihenfolge gestartet und der Cluster funktioniert.

ZUSATZ 1: FAILOVER TEST

Anschließend wurde ein Failover getestet. Zunächst eine Ausgabe mit „crm_mon -n“. ha1 ist Primary.

```
Stack: corosync
Current DC: ha2 (version 1.1.16-94ff4df) - partition with quorum
Last updated: Mon Oct 30 15:41:33 2017
Last change: Mon Oct 30 15:14:56 2017 by root via cibadmin on ha1

2 nodes configured
4 resources configured

Node ha1: online
    drbd_r0 (ocf::linbit:drbd):      Master
    filesystem (ocf::heartbeat:Filesystem):  Started
    clusterIP      (ocf::heartbeat:IPaddr):      Started
Node ha2: online
    drbd_r0 (ocf::linbit:drbd):      Slave
```

Nachdem ha1 gekillt wurde, sind Ressourcen von ha1 auf ha2 rüber gesprungen.

```
Stack: corosync
Current DC: ha2 (version 1.1.16-94ff4df) - partition with quorum
Last updated: Mon Oct 30 15:42:13 2017
Last change: Mon Oct 30 15:14:56 2017 by root via cibadmin on ha1

2 nodes configured
4 resources configured

Node ha1: OFFLINE
Node ha2: online
    drbd_r0 (ocf::linbit:drbd):      Master
    filesystem (ocf::heartbeat:Filesystem):  Started
    clusterIP      (ocf::heartbeat:IPaddr):      Started
```

Abschließend wurde ha1 wieder hochgefahren. Dieser wurde nach ca. 20sek automatisch zum Primary und alle Ressourcen sind vom ha2 wieder zurückgesprungen.

```
Stack: corosync
Current DC: ha2 (version 1.1.16-94ff4df) - partition with quorum
Last updated: Mon Oct 30 15:43:08 2017
Last change: Mon Oct 30 15:14:56 2017 by root via cibadmin on ha1

2 nodes configured
4 resources configured

Node ha1: online
    drbd_r0 (ocf::linbit:drbd):      Master
    filesys (ocf::heartbeat:Filesystem):  Started
    clusterIP      (ocf::heartbeat:IPaddr):      Started
Node ha2: online
    drbd_r0 (ocf::linbit:drbd):      Slave
```