KURZANLEITUNG: DOKUMENTATION_KONFIGURATION_DRBD_PP_V7

Projekt:	Intern	Erster Stand:	05.10.2017
Herausgeber:	Hellberg EDV Beratung	Letzter Stand:	07.12.2017
Alle Bearbeiter:	GH, PP	Letzter Bearbeiter:	GH

Dokumentation:
Kurzbeschreibung: Konfiguration von DRBD auf einem Debian 9.1.

Anmerkungen zur Dokumentation:

AUSGANGSLAGE / ZIELSETZUNG:

Für ein Projekt soll überprüft werden, ob die Dienste DRBD, Corosync, Pacemaker und NFS einwandfrei auf der Distribution Debian 9.1 laufen.

Versionen:

DRBD	8.9.10-2
Corosync	2.4.2-3
Pacemaker	1.1.16-1
NFS-Kernel	1.3.4-2.1

In dieser Dokumentation wird das DRBD konfigurieren und getestet.

Es wird vorausgesetzt, dass die beiden VM‘s (ha1 und ha2) bereits eine neue zweite Platte (/dev/sdb) haben, und diese bereits mit fdisk partitioniert wurden.

SCHRITT 1:

Vor der Konfiguration muss sichergestellt werden, ob der DRBD-Dienst aus ist. Gegebenenfalls muss dieser ausgeschaltet werden.

```
/etc/init.d/drbd stop
```

Alternativ kann auch folgender Befehl verwendet werden.

```
service drbd start
```

SCHRITT 2:

Als erstes wird eine Sicherheitskopie von der Datei "global_common.conf" angelegt. Im Falle eines Konfigurationsfehlers, kann diese Kopie wieder verwendet werden. Diese Konfigurationsdatei liegt im Verzeichnis etc/drbd.d/. Die Sicherheitskopie wird auf beiden Knoten erstellt!

```
cp global_common.conf global_common.conf.orginal
```

SCHRITT 3:

Nun wird die Datei „global_common.conf“ konfiguriert. Diese muss auf beiden Knoten identisch sein!

```
global {
      usage-count no;
}
common {
}
resource r0 {
      protocol C;
      disk {
      on-io-error pass_on;
      resync-rate 100M;
      }
      on ha1 {
      device /dev/drbd0;
      address 4.4.4.8:7789;
      meta-disk internal;
      disk /dev/sdb1;
      }
      on ha2 {
      device /dev/drbd0;
      address 4.4.4.9:7789;
      meta-disk internal;
      disk /dev/sdb1;
      }
}
```

In der Sektion global wird mit "usage-count no" angegeben, dass eine Teilnahme an Linbits Onlinezählung nicht erwünscht ist.

Alle in der "common" Sektion vorgenommenen Einstellungen, werden auf alle Ressource vererbt. Dies dient zur einer besseren Übersicht bei multipler DRBD-Ressourcen Verwaltung. Da wir momentan nur eine Ressource haben, bleibt diese Sektion vorerst leer.

In der Sektion "resource" werden alle Eigenschaften der DRBD-Ressource definiert. In diesem Fall, werden die beiden Knoten später über das Protokoll C synchronisiert. In der Subsektion "disk" wird der Synchronisation Wert angegeben, spricht welche Bandbreite DRBD nutzen darf. Dieser Wert ist insbesondere dann wichtig, wenn auch andere Prozesse über diese Leitung Daten austauschen sollten, dann kann dieser Wert angepasst werden, damit die Prozesse nicht völlig ausgebremst werden.

Die beiden Host-Sektionen "on hostname" legen die eigentliche DRBD-Legs fest. Zu den erforderlichen Werten gehört der jeweilige Hostname des Knotens (ha1 bzw. ha2), die dazu gehörige IP und der Port, über den die DRBD-Kommunikation abgewickelt wird, der Name des Blockdivices (dev/drbd0) und die dem DRBD unterliegende Partition (/dev/sdb1).

Anmerkung: Die Sektion „resource r0“ kann auch in eine separate .res Datei ausgelagert werden.

SCHRITT 4:

Jetzt kann das DRBD mit neuen Metadaten initialisiert werden, und zwar **auf beiden Knoten**! Die Ressource soll in diesem Fall „r0“ heißen. Die Informationen für die Ressource r0 werden aus der global_common.conf entnommen, die vorhin Konfiguriert wurde.

```
drbdadm create-md r0
```

Mit „yes“ bestätigen.

```
root@ha1:/etc/drbd.d# drbdadm create-md r0
initializing activity log
NOT initializing bitmap
Writing meta data...
New drbd meta data block successfully created.
root@ha1:/etc/drbd.d#
```

```
root@ha2:/etc/drbd.d# drbdadm create-md r0
initializing activity log
NOT initializing bitmap
Writing meta data...
New drbd meta data block successfully created.
root@ha2:/etc/drbd.d#
```

SCHRITT 5:

Auf beiden Knoten wird der Dienst gestartet.

```
/etc/init.d/drbd start
```

Alternativ kann auch folgender Befehl verwendet werden.

```
service drbd start
```

SCHRITT 6:

Weiter geht’s mit dem **ha1** Knoten. Wenn die DRBD Platten neu erstellt wurden, und darauf noch nie Daten gewesen sind, dann kann mit dem folgenden Befehl angegeben werden, dass die beiden DRBD-Legs "clean" und "consistetnt" sind. Dadurch muss nicht unnötig gewartet werden, bis sich die beiden DRBD-Legs vollständig synchronisiert haben.

```
drbdadm -- --clear-bitmap new-current-uuid r0
```

SCHRITT 7:

Anschließend wird dieser Knoten (**ha1**) manuell auf primary gesetzt.

```
drbdadm primary r0
```

SCHRITT 8:

Mit dem Befehl cat /proc/drbd sehen wir jetzt, dass ha1 der primary ist und die beiden Datenbestände auf beiden Knoten auf UpToDate sind.

```
root@ha1:/etc/drbd.d# cat /proc/drbd
version: 8.4.7 (api:1/proto:86-101)
srcversion: 0904DF2CCF7283ACE07D07A
 0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----
    ns:0 nr:0 dw:0 dr:0 al:8 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
root@ha1:/etc/drbd.d#
```

SCHRITT 9:

Auf dem Knoten, der den DRBD-Primary hält, wird ein Dateisystem erzeigt. In diesem Fall wurde ext4 genommen.

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

```
root@HA1:/etc/drbd.d# mkfs.ext4 /dev/drbd0
mke2fs 1.43.4 (31-Jan-2017)
Ein Dateisystems mit 3798915 (4k) Blöcken und 950272 Inodes wird erzeugt.
UUID des Dateisystems: d28aad1b-cd60-450f-b308-22aaf4a5f761
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@HA1:/etc/drbd.d#
```

SCHRITT 10:

Jetzt wird auf **beiden Knoten** ein Ordner erstellt, wo das logische Gerät später eingehängt wird. Hier wurde der Name /daten gewählt.

```
mkdir /daten
```

SCHRITT 11:

Nun kann das logische Gerät **auf dem Primary** Knoten eingehängt werden.

```
mount /dev/drbd0 /daten
```

SCHRITT 12:

Mit dem Befehl *df –h* sehen wir, dass das logischer Gerät */dev/drbd0* nun auf */daten* eingehängt ist.

```
root@HA1:/# df -h
Dateisystem    Größe Benutzt Verf. Verw% Eingehängt auf
udev            992M       0  992M    0% /dev
tmpfs           201M    6,5M  195M    4% /run
/dev/sda3        19G    3,8G   14G   23% /
tmpfs          1003M       0 1003M    0% /dev/shm
tmpfs             5M    4,0K  5,0M    1% /run/lock
tmpfs          1003M       0 1003M    0% /sys/fs/cgroup
/dev/sda4        29G     45M   27G    1% /data
/dev/sda1       922M     36M  823M    5% /boot
tmpfs           201M     28K  201M    1% /run/user/116
tmpfs           201M     28K  201M    1% /run/user/1000
/dev/sr0        3,6G    3,6G     0  100% /media/cdrom0
/dev/drbd0       15G     41M   14G    1% /daten
root@HA1:/#
```

SCHRITT 13: MANUELLER DRBD-FUNKTIONSTEST

Jetzt wird die Funktionsweise getestet. Dafür wird ein manueller Failover durchgeführt. Es wird davon ausgegangen, dass das logische Gerät auf */daten* eingehängt ist und der Zustand der beiden Knoten sich auf *Primary/Secondary* und *UpToDate/UpToDate* befindet.

Zunächst wird eine Testdatei im Verzeichnis */daten* erstellt.

```
touch test
```

```
root@HA1:/daten# touch test
root@HA1:/daten#
```

SCHRITT 14:

Anschließend wird sie mit einem Editor geöffnet und beschrieben.

```
  GNU nano 2.7.4                    Datei: test

Dies ist ein Test auf HA1.
```

SCHRITT 15:

Jetzt wird der Montpunkt /daten auf dem Primary wieder ausgehangen.

```
umount /daten
```

SCHRITT 16:

Und der Primary wird zum Secondary Herabgestuft.

```
Drbdadm secondary r0
```

cat /proc/drbd sollte jetzt den Zustand *Secondary/Secondary* anzeigen.

SCHRITT 17:

Jetzt wechseln wir auf den HA2 Knoten und stufen diesen zum Primary auf.

```
drbdadm primary r0
```

```
root@HA2:/# drbdadm primary r0
root@HA2:/#
```

SCHRITT 18:

Einhängen des DRBD-Devices auf dem aktuellen Primary (HA2).

```
mount /dev/drbd0 /daten
```

```
root@HA2:/# mount /dev/drbd0 /daten
root@HA2:/#
```

SCHRITT 19:

Abschließend wechseln wir in das Verzeichnis /daten.

```
cd /daten
```

```
root@HA2:/# cd /daten
root@HA2:/daten#
```

SCHRITT 20:

Jetzt können wir mit einem Editor die Testdatei weiter bearbeiten.

```
  GNU nano 2.7.4                    Datei: test

Dies ist ein Test auf HA1.

Nun können wir die Testdatei weiter bearbeiten.
```

Der Funktionstest wurde erfolgreich durchgeführt. All diese Schritte wird später die Cluster-Ressource übernehmen.