Hochverfügbare Virtualisierungsumgebung mit Red Hat Enterprise Virtualization und Distributed Replicated Block Device

Lukas Grimm
Puzzle ITC
grimm@puzzle.ch

6. März 2014

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Dieses Dokument entstand im Rahmen der Abschlussarbeit meiner Informatiker-Lehre und ist Bestandteil der Projekt-Dokumentation. Dieses Dokument basiert auf "High-Availability oVirt-Cluster with iSCSI-Storage"von Benjamin Alfery und Philipp Richter, 2013 LINBIT HA-Solutions GmbH.

1.1 Ziel dieser Anleitung

Diese Anleitung soll Schritt für Schritt aufzeigen, wie man aus zwei physikalischen Maschinen eine hochverfügbare Virtualisierungsumgebung aufbauen kann. Diese basiert auf einer virtuellen Maschine, auf welcher der RHEV-Manager läuft und einem iSCSI-Target als Speicher für die virtuellen Maschinen des RHEVs. Die beiden Maschinen dienen gleichzeitig als RHEV-Hypervisoren. Für die Replizierung der Daten der KVM-Maschine und des iSCSI-Targets wird DRBD verwendet. Für das Cluster-Management wird Pacemaker und Heartbeat eingesetzt.

1.2 Voraussetzungen

Diese Dokumentation geht von zwei physischen Maschinen aus, welche harhev1 und harhev2 genannt werden. Diese sind am lokalen Netz mit den IP-Adressen 10.1.0.6 und 10.1.0.7 sowie über einen Crossover-Link mit den IP-Adressen 192.168.10.10 und 192.168.10.11. Da RHEV bei den Hypervisor eine Bridge verwendet, macht es Sinn, die IP-Adressen für das lokale Netzwerk gleich über die Bridge rhevm (oder ovirtmgmt falls oVirt verwendet wird) laufen zu lassen.
DRBD wird die Nummern 0 (Ressourcenname: kvm-ovirtm) und 1 (Ressourcenname: iscsi) für die replizierten Volumen verwenden.
Dieses Dokument beschreibt eine RHEV/iSCSI/DRBD/Pacemaker Installation auf einer x86_64 Maschine mit Red Hat Enterprise Linux 6.5 mit Kernel Version 2.6.32-431.5.1.el6.x86_64 und DRBD Version 8.4.4.
Weiter wird davon ausgegangen, dass LVM verwendet wird. Auch wenn LVM nicht zwingend ist, wird LVM wegen seiner Flexibilität empfohlen.
Für die später gebrauchten Ressourcen sollten zwei LVs auf beiden Nodes erstellt werden:

```
1 lvcreate -L20G -n kvm_oVirtm system # für die virtuelle Maschine auf
    welcher der Manager läuft
2 lvcreate -L500G -n iscsi system # Storage für die virtuellen Maschinen
    welche im RHEV erstellt werden, Grösse entsprechend anpassen
```

Es wird davon ausgegangen, dass der Leser Linux, DRBD und Pacemaker Wissen besitzt.

2 Software

Es wird davon ausgegangen, dass die Systeme bereits aufgesetzt sind und am Red Hat Network angemeldet sind. Für ein Setup via Puppet ist zu beachten, dass für die Abhängigkeiten von Puppet der "RHEL Server Optional" Channel angehängt sein sollte.
Pacemaker ist ein Cluster Ressourcen Management Framework, welches gebraucht wird, um Ressourcen zu starten, zu stoppen, zu überwachen und zu migrieren. Es wird von Pacemaker $\geq$ 1.1.6 ausgegangen.
Heartbeat ist eine Cluster-Benachrichtigungs-Ebene, welche von Pacemaker verwendet wird. Es wird von Heartbeat $\geq$ 3.0.5 ausgegangen.
DRBD ist eine Netzwerkspeicherlösung welche verwendet wird um auf Blockdevice-Ebene Daten zu synchronisieren. Kompilierte Pakete sind im Linbit Repository enthalten. Es werden die Pakete drbd-utils und drbd-kmod gebraucht, welche die DRBD Administrationswerkzeuge und das Kernel Modul zur Verfügung stellen.
Libvirt ist ein opensource Management Tool für Virtualisierung. Libvirt liefert eine API zu verschiedenen Virtualisierungstechnologien wie KVM, QEMU, XEN und VMware ESX.
RHEV/RHEVM ist eine Management-Applikation für virtuelle Maschinen. Diese Anleitung wurde für die Version 3.3 erstellt, sollte jedoch auch mit 3.2 laufen.

2.1 Software Repositories

Ausgehend davon, dass das Betriebsystem komplett installiert und aktuell ist sowie die beiden Netzwerkinterfaces korrekt konfiguriert sind und funktionieren, ist der erste Schritt auf beiden Maschinen, die Linbit Repositories hinzuzufügen. Dies kann mit dem yum Puppet Modul[1] mit folgenden zusätzlichen Manifesten bewerkstelligt werden:

2.1.1 DRBD und Pacemaker Repositories

Linbit DRBD-Repository:

```
# linbit drbd repository
class yum::repos::linbit_drbd(
  $linbit_drbd_access_hash = <hash>
){
  yum::managed_yumrepo{
    'linbit_drbd':
    descr      => 'RHEL-6 - Linbit DRBD 8.4',
    baseurl    => "http://packages.linbit.com/${linbit_drbd_access_hash
    }/8.4/rhel6/x86_64",
    enabled    => 1,
    gpgcheck   => 0,
    priority   => 1;
  }
}
```

[1] http://puppet-modules.git.puzzle.ch/?p=module-yum.git;a=summary

Linbit Pacemaker-Repository:

```
# linbit drbd repository
class yum::repos::linbit_pacemaker(
  $linbit_access_hash = <hash>
){

  yum::managed_yumrepo{
    'linbit_pacemaker':
    descr    => 'RHEL-6 - Linbit Pacemaker',
    baseurl  => "http://packages.linbit.com/${linbit_access_hash}/pacemaker
    /rhel6/x86_64",
    enabled  => 1,
    gpgcheck => 0,
    priority => 1,
    exclude  => 'heartbeat, heartbeat-libs';
  }
}
```

Es ist sicherzustellen, dass der <hash> mit dem Hash, welcher von Linbit zur Verfügung gestellt werden, ersetzt wird. Falls man keinen Zugriff auf diesen hat, dann muss man die Installation von DRBD, pacemaker und heartbeat selber bewerkstelligen.

2.2 EPEL-Repository

Es wird empfohlen, das heartbeat Paket aus dem Epel-Repo zu installieren, da bei der Version heartbeat-3.0.4-1.el6.x86_64 aus dem Linbit Repo das init Skript Fehler aufweist[2].

[2] https://bugzilla.redhat.com/show_bug.cgi?id=1028127

3 DRBD

3.1 Installation

Für DRBD werden, wie bereits erwähnt, zwei Pakete gebraucht. Diese können mit *yum -y install drbd kmod-drbd* installiert werden. Da Pacemaker die Ressourcen verwalten soll, muss der DRBD-Service mit *chkconfig drbd off* deaktiviert werden.

3.2 Konfiguration

Die DRBD-Ressourcen müssen auf beiden Servern konfiguriert werden. Dies macht man, indem man unter */etc/drbd.d/* eine Datei mit dem Namen der Ressource und der Endung *.res* anlegt. Die erste Ressource */etc/drbd.d/kvm-ovirtm.res* sieht wie folgt aus:

```
resource kvm-ovirtm {
  net {
    protocol C;
  }
  volume 0 {
    device      minor 0;
    disk        /dev/system/kvm_oVirtm;
    meta-disk   internal;
  }
  on harhev1 {
    address      192.168.10.10:7788;
  }
  on harhev2 {
    address      192.168.10.11:7788;
  }
}
```

Analog dazu */etc/drbd.d/iscsi.res*:

```
resource iscsi {
  net {
    protocol C;
  }
  volume 0 {
    device      minor 1;
    disk        /dev/system/iscsi;
    meta-disk   internal;
  }
  on harhev1 {
    address     192.168.10.10:7789;
  }
  on harhev2 {
    address     192.168.10.11:7789;
  }
}
```

Anschliessend müssen die Ressourcen mit *drbdadm create-md $resource* erstellt werden und mit *drbdadm up $resource* aktiviert werden. Dies muss auf beiden Servern ausgeführt werden. Abschliessend muss ein Server als Primary gesetzt werden, dies geschieht mit *drbdadm primary –force $resource*. Den DRBD-Status kann mit *cat /proc/drbd* überprüft werden.

3.3 Installation und Konfiguration via Puppet

Die Installation und Konfiguration via Puppet kann mit folgenden Puppetmanifesten bewerkstelligt werden, welche auch bald unter `http://puppet-modules.git.puzzle.ch/` verfügbar sein werden. Die Aktivierung muss jedoch von Hand vorgenommen werden, damit Puppet nicht eine aktive Ressource zerstört.

3.3.1 harhev.pp

```
class drbd::harhev {
  include drbd::cluster
  include drbd::linbit

  /*
   stuff actually done manually:

   drbdadm create-md resource
   drbdadm up resource
   drbdadm primary --force resource
   */

  file {
    '/etc/drbd.d/kvm-ovirtm.res':
      source => "puppet:///modules/drbd/harhev/kvm-ovirtm.res",
      owner  => 'root', group => 0, mode => 0640;
    '/etc/drbd.d/iscsi.res':
      source => "puppet:///modules/drbd/harhev/iscsi.res",
      owner  => 'root', group => 0, mode => 0640;
  }
}
```

Diese Klasse legt die Konfigurationsdateien an, deren Inhalt oben beschrieben ist und deshalb hier nicht nochmal aufgeführt werden. Weiter werden zwei weitere Klassen eingebunden.

3.3.2 cluster.pp

```
class drbd::cluster {
  # ensure that drbd is not enabled as startup service because drbd service
     is controlled by rgmanager/pacemaker
   service { drbd:
     enable  => false,
     require => Package[drbd],
   }
```

Die Klasse Cluster stellt sicher, dass der DRBD-Service in einem Cluster nicht läuft, da diese Aufgabe von einem Clustermanagement-Tool wie Pacemaker übernommen wird.

3.3.3 linbit.pp

```
# class for servers which use packages from linbit
class drbd::linbit {
    include yum::repos::linbit_drbd

   package { "drbd":
     ensure => present,
   }

   package { "kmod-drbd":
     ensure => present,
     alias  => "drbd-module",
   }
}
```

Diese Klasse bindet die zuerst die Repo-Klasse, welche unter Software deklariert worden ist, ein und installiert anschliessend die notwendigen Pakete.

4 KVM Maschine definieren

Auf beiden Server muss eine virtuelle Maschine erstellt werden, welche die *kvm-ovirtm*-DRBD-Ressource als Harddisk eingebunden hat. Die Definition sollte *ähnlich* der folgenden sein:

```
<domain type='kvm'>
  <name>oVirtm</name>
  <uuid>34ad3032-f68e-734a-8e84-47af69e7848a</uuid>
  <memory unit='KiB'>6291456</memory>
  <currentMemory unit='KiB'>6291456</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <os>
    <type arch='x86_64' machine='rhel6.4.0'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
```

```
14      <pae/>
15    </features>
16    <clock offset='utc'/>
17    <on_poweroff>destroy</on_poweroff>
18    <on_reboot>restart</on_reboot>
19    <on_crash>restart</on_crash>
20    <devices>
21      <emulator>/usr/libexec/qemu-kvm</emulator>
22        <disk type='block' device='disk'>
23          <driver name='qemu' type='raw' cache='none'/>
24          <source dev='/dev/drbd/by-res/kvm-ovirtm/0'/>
25          <target dev='vda' bus='virtio'/>
26          <address type='pci' domain='0x0000' bus='0x00' slot='0x04'/>
27        </disk>
28      <interface type='bridge'>
29        <mac address='52:54:00:08:10:da'/>
30        <source bridge='ovirtmgmt'/>
31        <model type='virtio'/>
32        <address type='pci' domain='0x0000' bus='0x00' slot='0x03'/>
33      </interface>
34      <serial type='pty'>
35        <target port='0'/>
36      </serial>
37      <console type='pty'>
38        <target type='serial' port='0'/>
39      </console>
40      <input type='tablet' bus='usb'/>
41      <input type='mouse' bus='ps2'/>
42      <graphics type='vnc' port='-1' autoport='yes'/>
43      <video>
44        <model type='cirrus' vram='9216' heads='1'/>
45        <address type='pci' domain='0x0000' bus='0x00' slot='0x02'/>
46      </video>
47      <memballoon model='virtio'>
48        <address type='pci' domain='0x0000' bus='0x00' slot='0x05'/>
49      </memballoon>
50    </devices>
51  </domain>
```

Es wird empfohlen, die Maschine gleich mit einem RHEL 6.5 aufzusetzen, alle weiteren Schritte zur Installation folgen später.

5 Pacemaker und Heartbeat

5.1 Installation

Für Pacemaker werden die Pakete *drbd-pacemaker pacemaker-hb pacemaker-hb-cli heartbeat* gebraucht. Wie bereits beschrieben, wird die heartbeat am besten aus dem EPEL-Repo installiert, da bei dieser Version der init-Script Bug gefixt wurde.

5.2 Konfiguration Heartbeat

Um die Clusterkommunikation zu ermöglichen, muss Heartbeat konfiguriert werden. Unter */etc/ha.d/ha.cf* sind die Heartbeatparameter abgelegt. Diese sollten in etwa so aussehen:

```
autojoin none
node harhev1
node harhev2
bcast eth1
mcast rhevm 239.192.0.61 694 1 0
use_logd yes
initdead 120
deadtime 20
warntime 10
keepalive 1
compression bz2
crm respawn
```

Weiter sollte im selben Verzeichnis eine Datei mit dem Namen *authkeys* angelegt werden, welche in etwa so aussieht:

```
auth 1
1 sha1 sdrsdfrgaqerbqerbq34bgaebaqejrbnSDFQ23Fwe
```

Dieser Key wird als Shared-Secret für die Cluster-Kommunikation verwendet. Es ist sicherzustellen, dass diese Datei die richtigen Berechtigungen hat:

```
chown root: /etc/ha.d/authkeys
chmod 600 /etc/ha.d/authkeys
```

Anschliessend kann Heartbeat mit *service heartbeat start* gestartet werden. Es sollte sichergestellt werden, dass Heartbeat beim Systemstart gestartet wird: *chkconfig heartbeat on*.

5.3 Allgemeine Pacemakerregeln

Als erstes sollten allgemeine Clusterkonfigurationen definiert werden. Diese können über die *crm*-Shell auf einem Server getätigt werden. Es ist zu beachten, dass die Timeout-Zeiten entsprechend den Bedürfnissen und der Hardware angepasst werden sollten:

```
property $id="cib-bootstrap-options" \
  no-quorum-policy="ignore" \
  cluster-recheck-interval="1min"

rsc_defaults $id="rsc-options" \
   resource-stickiness="200" \
   migration-threshold=2 \
   failure-timeout=300s
```

Weiter sollte STONITH (Shoot the other node in the head) konfiguriert werden, damit im Fall eines Fehlers der Server neu gestartet werden kann. Die hier beschriebene Konfiguration geht davon aus, dass die Server ein IPMI (mit den IP-Adressen 10.1.0.4 und 10.1.0.5) besitzen:

```
primitive st_harhev1 stonith:external/ipmi \
   params hostname="harhev1" ipaddr="10.1.0.4" userid="ADMIN" passwd="
    password" interface="lanplus"
   location l_st_harhev1 st_harhev1 -inf: harhev1

primitive st_harhev2 stonith:external/ipmi \
   params hostname="harhev2" ipaddr="10.1.0.5" userid="ADMIN" passwd="
    password" interface="lanplus"
   location l_st_harhev2 st_harhev2 -inf: harhev2
```

Durch einen Fehler in Heartbeat wird hier eine Fehlermeldung auftreten: *WARNING: l_st_harhev2: referenced node harhev2 does not exist*. Dies liegt daran, dass Heartbeat die Node-ID mit dem Node-Namen vergleicht, anstatt den beiden Namen. Diese Meldung kann ignoriert werden.
Die vorgenommenen Änderungen sollten zuerst in der *crm*-Shell getestet werden und anschliessend mit *commit* scharf geschaltet werden.

5.4 Pacemakerregeln für den Manager

Als nächstes sollte die DRBD-Ressource für den Manager in Pacemaker konfiguriert werden:

```
primitive p_drbd_kvm-ovirtm ocf:linbit:drbd \
  params drbd_resource="kvm-ovirtm" \
  op monitor interval="29" role="Master" timeout="30" \
  op monitor interval="30" role="Slave" timeout="30" \
  op start interval="0" timeout="240" \
  op stop interval="0" timeout="100"
```

Da diese Ressource über zwei Server spannt, braucht es ein Master/Slave-statement:

```
ms ms_drbd_kvm-ovirtm p_drbd_kvm-ovirtm \
  meta clone-node-max="1" clone-max="2" master-max="1" master-node-max="1"
    notify="true"
```

Als nächstes definieren wir die virtuelle Maschine:

```
primitive p_kvm-ovirtm ocf:heartbeat:VirtualDomain \
params config="/etc/libvirt/qemu/oVirtm.xml" \
op start interval="0" timeout="180s" \
op stop interval="0" timeout="300s" \
op monitor interval="60s" timeout="60s"
```

Da die Disk und die virtuelle Maschine auf demselben Server laufen sollen müssen noch zwei Einschränkungen definieren, die erste definiert, dass die beiden Ressourcen auf dem gleichen Server laufen:

```
colocation co_kvm-ovirtm_with_drbd +inf: p_kvm-ovirtm:Started ms_drbd_kvm-
    ovirtm:Master
```

Die zweite definiert die Reihenfolge, damit die Maschine erst gestartet wird, wenn die Disk bereit ist:

```
order o_drbd-kvm-ovirtm_before_kvm +inf: ms_drbd_kvm-ovirtm:promote p_kvm-
    ovirtm:start
```

Auch hier gilt es, die Änderungen zuerst zu überprüfen und dann zu commiten.

6 Pacemakerregeln für iSCSI

Als erstes muss auch für die iSCSI die DRBD-Ressource definiert werden:

```
primitive p_drbd_iscsi ocf:linbit:drbd \
  params drbd_resource="iscsi" \
  op monitor interval="29" role="Master" timeout="30" \
  op monitor interval="30" role="Slave" timeout="30" \
  op start interval="0" timeout="240" \
  op stop interval="0" timeout="100"
```

Da sich auch diese Ressource über zwei Server spannt, braucht es auch hier wieder ein Master/Slave-statement:

```
ms ms_drbd_iscsi p_drbd_iscsi \
  meta clone-node-max="1" clone-max="2" master-max="1" master-node-max="1"
    notify="true"
```

Als nächstes wird das iSCSI-Target definiert:

```
primitive p_iscsi_store1 ocf:heartbeat:iSCSITarget \
  params implementation="tgt" iqn="iqn.2014-02-14.linbit.ovirtiscsi:store1"
     tid="1" \
  op start interval="0" timeout="60" \
  op stop interval="0" timeout="60" \
  op monitor interval="30" timeout="60"
```

Dieses Target wird nun ein Lun (logical unit) zugewiesen, mit der DRBD-Ressource als Backingdevice:

```
primitive p_iscsi_store1_lun1 ocf:heartbeat:iSCSILogicalUnit \
  params implementation="tgt" target_iqn="iqn.2014-02-14.linbit.ovirtiscsi:
    store1" lun="1" \
  path="/dev/drbd/by-res/iscsi/0" \
  op start interval="0" timeout="60" \
  op stop interval="0" timeout="60" \
  op monitor interval="30" timeout="60"
```

Um vom Server unabhängig auf das definierte Target zuzugreifen, definieren wir eine Service-IP-Adresse:

```
primitive p_ip_iscsi ocf:heartbeat:IPaddr2 \
  params ip="192.168.10.50" \
  op start interval="0" timeout="20" \
  op stop interval="0" timeout="20" \
  op monitor interval="30" timeout="20"
```

Da bei es bei einem Switch- oder Failover zu tcp-Reject Meldungen kommen kann, wird ein Portblock gesetzt, welcher während einem Switch alle Pakete am iSCSI-Port dropt. Dies ist die sicherere Variante, da dies von RHEV als Latenz-Problem verstanden wird und nicht als Fehler:

```
primitive p_portblock-store1-block ocf:heartbeat:portblock \
  params ip="192.168.10.50" portno="3260" protocol="tcp" action="block"
```

Dieser Block muss auch wieder aufgehoben werden:

```
primitive p_portblock-store1-unblock ocf:heartbeat:portblock \
  params ip="192.168.10.50" portno="3260" protocol="tcp" action="unblock" \
  op monitor interval="30s"
```

Da diese iSCSI-Primitives immer zusammen laufen sollen, müssen sie gruppiert werden:

```
group g_iscsi p_portblock-store1-block p_ip_iscsi p_iscsi_store1 \
  p_iscsi_store1_lun1 p_portblock-store1-unblock
```

Schlussendlich müssen wiederum zwei Einschränkungen definiert werden, damit sowohl die DRBD-Ressource wie auch die Dienste auf dem selben Server laufen:

```
colocation co_g_iscsi_with_drbd +inf: g_iscsi:Started ms_drbd_iscsi:Master
```

Und damit die iSCSI-Dienste erst gestartet werden, wenn die DRBD-Ressource bereit ist:

```
order o_drbd_iscsi_before_g_iscsi +inf: ms_drbd_iscsi:promote g_iscsi:start
```

6.1 Pacemaker und Puppet

Puzzle ITC ist gerade daran, ein Puppet-Modul zu Pacemaker zu schreiben. Durch den Fehler in Heartbeat, wie oben beschrieben, ist es leider noch nicht möglich, die Konfiguration via Puppet einzuspielen. Sobald dies möglich ist, wird das Modul unter `http://puppet-modules.git.puzzle.ch/` veröffentlicht und diese Anleitung entsprechend aktualisiert.

7 RHEV Installation

Die Installation von RHEV wird hier aus verschiedenen Gründen nicht mit Puppet beschrieben.

7.1 Installation RHEV-Manager

Für die Installation des RHEV-Manager wurde im Kapitel "KVM Maschine Definieren" bereits die virtuelle Maschine erstellt. Dieser Installation müssen folgende Channels angehängt werden:

- RHEL Server Optional
- RHEL Server Supplementary
- Red Hat Enterprise Virtualization Manager (v.3.3 x86_64)
- Red Hat JBoss EAP (v 6) for 6Server x86_64

Anschliessend kann mit *engine-setup* das Managerpaket installiert werden. Vor dem Setup der Managementengine sollte sichergestellt werden, dass die Maschine eine feste IP-Adresse hat und der DNS-Name korrekt aufgelöst wird; ansonsten wird das Setup nicht durchlaufen. Anschliessend kann mit *engine-setup* die Manager-Konfiguration gestartet werden. Mit *−−config=* kann eine Konfigurationsdatei eingespielt werden, ein Beispiel ist hier aufgeführt:

```
# action=setup
[environment:default]
OVESETUP_CORE/engineStop=none:None
OVESETUP_DIALOG/confirmSettings=bool:True
OVESETUP_DB/database=str:engine
OVESETUP_DB/fixDbViolations=none:None
OVESETUP_DB/secured=bool:False
OVESETUP_DB/host=str:localhost
OVESETUP_DB/user=str:engine
OVESETUP_DB/securedHostValidation=bool:False
OVESETUP_DB/password=str:your_db_password_here
OVESETUP_DB/port=int:5432
OVESETUP_SYSTEM/nfsConfigEnabled=bool:False
OVESETUP_SYSTEM/memCheckEnabled=bool:True
OVESETUP_PKI/organization=str:example.com
OVESETUP_CONFIG/isoDomainName=none:None
OVESETUP_CONFIG/adminPassword=str:your_admin_password_here
OVESETUP_CONFIG/applicationMode=str:both
OVESETUP_CONFIG/firewallManager=str:iptables
OVESETUP_CONFIG/updateFirewall=bool:True
OVESETUP_CONFIG/websocketProxyConfig=bool:True
OVESETUP_CONFIG/fqdn=str:manager.example.com
OVESETUP_CONFIG/isoDomainMountPoint=none:None
OVESETUP_CONFIG/storageType=str:iscsi
OVESETUP_PROVISIONING/postgresProvisioningEnabled=bool:True
OVESETUP_APACHE/configureRootRedirection=bool:True
```

```
OVESETUP_APACHE/configureSsl=bool:True
OSETUP_RPMDISTRO/requireRollback=none:None
OSETUP_RPMDISTRO/enableUpgrade=none:None
OVESETUP_AIO/configure=none:None
OVESETUP_AIO/storageDomainDir=none:None
OVESETUP_DIALOG/confirmUpgrade=bool:True
OVESETUP_SYSTEM/redhatSupportProxyPort=none:None
OVESETUP_SYSTEM/redhatSupportProxy=none:None
OVESETUP_SYSTEM/redhatSupportProxyUser=none:None
OVESETUP_SYSTEM/configureRedhatSupportPlugin=bool:False
OVESETUP_SYSTEM/redhatSupportProxyPassword=none:None
OVESETUP_SYSTEM/redhatSupportProxyEnabled=bool:False
```

7.2 LVM-Rekonfiguration

Bevor die Hypervisoren in die Engine eingebunden werden, müssen zuerst ein paar Anpassungen an der LVM-Konfiguration vorgenommen werden, da auch der Hypervisor LVM verwenden wird. Diese Anpassungen müssen auf beiden physischen Systemen angewendet werden.
In */etc/lvm/lvm.conf* muss *write_cache_state=0* gesetzt werden. Beim preferred_names Parameter muss der Name der Volume Group hinzugefügt werden, wenn der Name der Volume Group system ist, sieht dies wie folgt aus:

```
preferred_names = [ "^/dev/mpath/", "^/dev/mapper/mpath", ... , "^/dev/
    system" ]
```

Schlussendlich muss noch der Filter um DRBD-Devices erweitert werden:

```
filter = [ "r|^/dev/drbd.*|" ]
```

7.3 Erweiterung zur udev-Regel für DRBD

Da die Installation der Hypervisor die qemu-Konfiguration ändern wird und pacemaker auf DRBD zugreifen muss, sollte die DRBD-udev-Regel wiefolgt angepasst werden:

```
SUBSYSTEM=="block", KERNEL=="drbd*", IMPORT{program}="/sbin/drbdadm sh-udev
    minor-%m", NAME="$env{DEVICE}", SYMLINK="drbd/by-res/$env{RESOURCE}
    drbd/by-disk/$env{DISK}", OWNER="vdsm", GROUP="kvm"
```

7.4 Hypervisor Installation

Damit die beiden physischen Server als Hypervisoren verwendet werden können, müssen noch die entsprechenden Pakete installiert werden. Dies wird vom Manager erledigt. Zuerst muss noch der Channel "Red Hat Enterprise Virt Management Agent (v.6 for x86_64)" angehängt werden.
Danach muss der Server für pacemaker in "standby" versetzt werden, via crm node standby hostname. Nachdem alle Ressourcen auf dem anderen Host laufen, kann im Webinterface der Hypervisor hinzugefügt werden. Wenn man sich im Administrations-Interface einloggt,

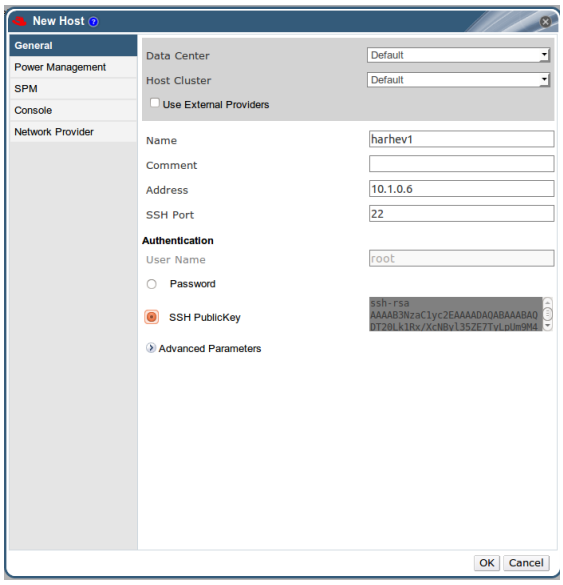

Abbildung 1: Hinzufügen eines Hosts in RHEV

kann man unter dem Tab "Hosts" den Button "new" klicken und die Angaben zum neuen Host eingeben. Sobald man "ok" klickt, wird der Host installiert. Im unteren Bereich oder im "Events" Tab kann man die Istallation verfolgen. Sobald die Installation abgeschlossen wurde, wird der Host neu gestartet.

7.4.1 RHEV-Powermanagement

RHEV fragt nach, ob man das Powermanagement für den neuen Host konfigurieren will oder nicht. Dieses Feature sollte im Zusammenhang mit Pacemaker mit Vorsicht genossen werden. Das Fencing von Pacemaker und das von RHEV basiert auf komplett unterschiedlichen Kriterien. Diese Konfiguration könnte, falls man keine weitere Anpassungen vornimmt, den Cluster in einen irreparablen Zustand versetzen.

7.5 Anpassungen an libvirt

Die RHEV-Installation sichert den Zugriff auf libvirt mit einem Passwort. Jedoch muss für die virtuelle Maschine, auf der der Manager läuft, Zugriff auf libvirt gewährleistet sein. Dies kann erreicht werden, indem für Pacemaker ein Passwort gesetzt wird. Dies muss mit *saslpasswd2 -a libvirt pcmk* erstellt werden und dann unter /etc/libvirt/auth.conf abgelegt werden:

```
1 [credentials-pcmk]
2 authname=pcmk
3 password=your-password-from-saslpasswd2
4 [auth-libvirt-ovirt-hyp1]
5 credentials=pcmk
6 [auth-libvirt-ovirt-hyp2]
7 credentials=pcmk
8 [auth-libvirt-localhost]
9 credentials=pcmk
```

Ob die Authentifizierung funktioniert, kann zum Beispiel mit *virsh list* getestet werden; falls kein Passwort gefragt wird, funktioniert diese wie gewünscht.

7.6 Zweite Hypervisor Installation

Für die Installation des zweiten Hypervisors müssen zuerst wiederum alle Ressourcen verschoben werden. Es muss sichergestellt werden, dass beide DRBD-Ressourcen wieder synchronisiert sind. Danach kann der Host in "standby" gesetzt werden und anschliessend wiederum übers Management-Interface installiert werden.
Nach dem reboot ist wiederum das Passwort für libvirt zu setzen, danach kann der Host wieder online genommen werden.

8 Storage Installlation

Als nächster Schritt folgt das Einbinden des iSCSI-Storage ins RHEV. Auch dieser Schritt kann wieder über das Administrations-Portal erledigt werden. Unter dem Tab "Storage" kann via "New Domain" neuer Speicherplatz hinzugefügt werden. Für das Einbinden des iSCSI-Targets muss man zuerst einen Namen wählen. Danach kann unter "Adress" die IP-Adresse, welche als Pacemaker-Ressource definiert wurde, eingegeben werden. Mit "Discover" werden die verfügbaren Targets aufgelistet. Wenn man sich beim entsprechenden Target einloggt, kann die "LUN ID " ausgewählt werden. Anschliessend sollte die Aktion mit "OK" bestätigt werden.
Nach kurzer Zeit sollte die Storage-Domain auf "Active" wechseln. Somit sind alle relevanten Komponenten bereit zur Nutzung.

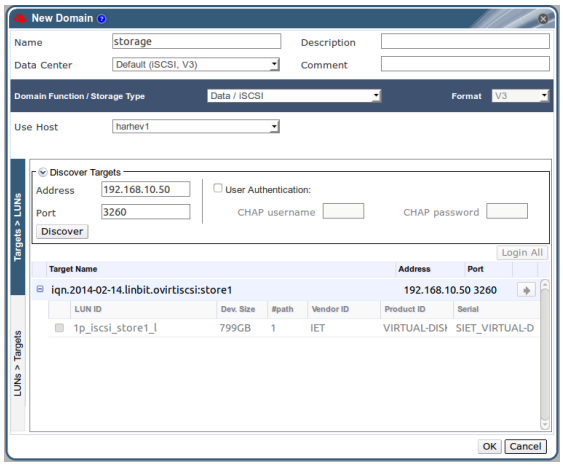

Abbildung 2: Hinzufügen von iSCSI-Storage in RHEV

9 Testen

Diese Anleitung beschreibt lediglich einen einfachen Aufbau eines High Availability Clusters mit RHEV. Dieser Aufbau ist keineswegs für produktive Umgebungen gedacht! Zuerst müssen entsprechende Tests, wie zum Beispiel des Fencens durchgeführt werden. Auch zu beachten ist, dass DRBD nicht als Backup missverstanden werden darf.