Oracle VM3: Virtuelle Maschinen per Script erstellen

Martin Bracher
Trivadis AG
Bern / Zürich

Schlüsselworte
OracleVM, XEN, Cloning, Template, OVMCLI, Kommandozeile, OVMD, ovmmessage, Enterprise Linux,

Einleitung

Mit den aktuellen OracleVM Versionen bietet Oracle nun die Möglichkeit, per Kommandozeile virtuelle Maschinen zu erstellen.

Ebenso hat Oracle eine neue Schnittstelle bereit gestellt (ovmd, ovm_template_config* RPM's), um aus einem Template hergestellte VM's interaktiv oder automatisch (per Messaging) zu konfigurieren, wie etwa Netzwerk oder Passwörter. Diese Schnittstelle lässt sich auch noch um eigene Module erweitern.

Dieser Vortrag entstand aus den Erfahrungen aus einem Kundenprojekt, in welchem wir von der Erstellung der VM bis zur Installation einer Datenbank (RAC/Data Guard) alles voll automatisiert erstellen konnten.

Themen:

- Wie erstellt man ein Template
- Wie funktionieren die Template-Konfigurationsscripts von Oracle
- Wie erstelle ich eigene Module, z.B. zum Aufbau eines RAC-Clusters
- Cloning einer VM aus einem Template
- Übergabe von Konfigurationsvariablen an die VM

Begriffe

Was ist eine Virtuelle Maschine?

Eine Virtuelle Maschine (VM) ist für den Benutzer, die Applikation wie ein physischer Server. Er sollte keinen Unterschied feststellen.

Das Betriebssystem und die Applikationen laufen auf virtueller „Hardware“. Diese virtuelle Hardware ist eine Zuordnung zu Teilen der physischen Hardware. Netzwerk und Teile des RAM und CPU werden vom Host mitbenutzt.

Auf dem Server ist dies in einem Konfigurationsfile vm.cfg hinterlegt.

```
bootloader = '/usr/bin/pygrub'
disk = ['file:/OVS/Repositories/004fb…/VirtualDisks/System.img,xvda,w',
        'file:/OVS/Repositories/004fb…/VirtualDisks/data.img,xvdb,w']
memory = '1024'
```

```
name = 'VM001'
vcpus = 1
on_crash = 'restart'
on_reboot = 'restart'
vfb = [type=vnc,vncunused=1,vnclisten=127.0.0.1,keymap=en-us]
vif = [mac=00:21:f6:99:73:1d,bridge=0004fb00108e6b2]
```

Die virtuellen Disks werden einem File oder einem Block-Device auf dem physischen Host zugewiesen.

```
-rw-r--r-- 1 root root  6448619520 Aug  7 17:36 System.img
-rw-r--r-- 1 root root 53694627840 Aug  7 17:36 data.img
```

Was ist Cloning?

Cloning ist das Kopieren einer bestehenden VM oder eines sogenannten Templates, und diese Kopie danach als neue VM zu verwenden. Dies läuft grundsätzlich in drei Schritten ab:

1. Definieren einer neuen VM (Mapping zur Hardware)
2. Kopieren der Disks der bestehenden VM/Template
3. Mapping der kopierten Disks zur neuen VM

Diese drei Schritte können im OVMCLI mit einem Befehl gemacht werden:

```
clone vm name=master1 destType=Vm destName=vm001 serverPool=myPool
         targetRepository=myRepo
```

Man kann die Schritte aber auch selbst einzeln durchführen:

```
create vm name=vm001 domainType=XEN_PVM …
create vnic  name=vm001_eth0 network=vlanpublic on vm name=vm001
cloneVdToRepo VirtualDisk name=MyDisk target=MyRepo cloneType=THIN_CLONE
create VmDiskMapping slot=0 VirtualDisk=… name=xvda on vm name=vm001
```

Die beiden Varianten lassen sich aber auch kombinieren. Man kann ein Template haben, das nur die minimal notwendigen Disks für das Betriebssystem besitzt, und Disks für die Daten, z.B. LUN's für ASM werden erst nach dem Cloning nach Bedarf hinzugefügt.

```
clone vm name=master_oel6 destType=Vm destName=vm201 serverPool=slot
         targetRepository=slotrepottcovms02sdb;

create VirtualDisk size=3.3 shareable=yes sparse=yes name=vm201vm202_xvdd
                  on repository name=slotrepottcovms02sdb;

create VmDiskMapping slot=3 VirtualDisk=vm201vm202_xvdd name=vm201_grid1
                     on vm name=vm201;
```

Was ist ein Template?

Das Problem beim Cloning ist, dass wir danach zwei identische VM's haben, also mit identischer IP-Adresse, identischem Hostname, identischen SSH-Keys, identischer up2date UUID.

Bevor wir die VM benützen können, müssen wir diese zuerst entsprechend konfigurieren. Genau diese Problematik adressiert ein Template.

Ein Template ist eine VM, welche erkennt, dass sie geklont wurde und sich beim ersten Booten neu konfiguriert.

Wie kann eine aus einem Template geklonte VM erkennen, dass sie noch nicht neu konfiguriert wurde?

Man kann den Ansatz von Oracle verwenden. In einem Konfigurationsfile wird die Variable RUN_TEMPLATE_CONF (alte Implementierung) oder INITIAL_CONFIG (neue Implementierung) auf ‚YES' gesetzt. Dann wird die VM neu konfiguriert und nach Abschluss der Konfiguration wird der Wert auf NO gesetzt, so dass beim nächsten Reboot nicht wieder eine Neu-Konfiguration stattfindet. Der Nachteil dieser Variante ist, dass man bei der Erstellung der Master-VM oder des Templates vor dem Stoppen den Wert ändern muss. Falls man das vergessen hat, kann man direkt auf dem Server mit folgendem Trick die Änderung noch nachholen:

```
kpartx -av <uuid.img>
mount /dev/mapper/loopXp2 /mnt; vi /mnt/sysconfig/ovmd; umount /mnt
kpartx -dv <uuid.img>
```

Ein anderer Ansatz ist, dass man auf der MAC-Adresse der virtuellen Netzwerkkarte basiert: Beim Klonen einer VM wird jeweils ein neues eth0 Interface mit einmaliger MAC-Adresse „eingebaut". Wenn sich also in /etc/sysconfig/network-scripts/ifcfg-eth0 eine andere MAC-Adresse steht, als das aktuelle Interface hat, kann man davon ausgehen, dass die VM gerade gecloned wurde. Zu beachten ist, dass Enterprise Linux Versionen > 5.7 das Konfigurationsfile automatisch durch eines mit dhcp-Konfiguration austauschen, falls die MAC-Adresse nicht mehr übereinstimmt. Also muss man sein Runlevel-Script sehr tief ansiedeln, oder wenn wir mit statischen Adressen arbeiten, können wir auf DHCP-Konfiguration prüfen.

Erstellung einer Master-VM oder eines Templates

Bevor wir mit dem Klonen beginnen können, brauchen wir ein Master-Template. Dieses können wir entweder von Grund auf neu erstellen, oder wir können eine bestehende VM oder Template anpassen.

Oracle-Templates

Oracle selbst bietet auch Templates an. Diese können unter https://edelivery.oracle.com/linux heruntergeladen werden.

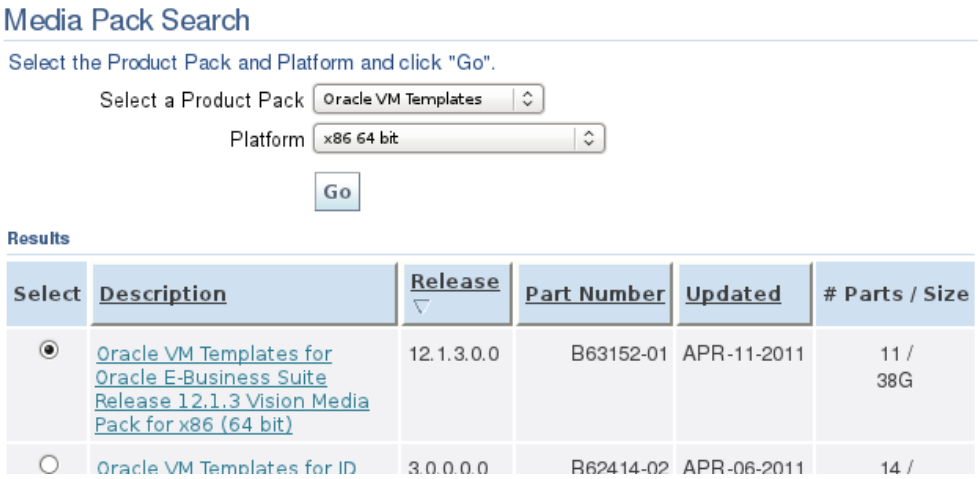

Abb. 1: Oracle edelivery Download-Bereich für Templates

Es stehen hier Templates für verschiedenste Zwecke zur Verfügung. Von der reinen Betriebssystem-Installation über Applikationen wie Enterprise-Manager oder Siebel bis zu Single- und RAC-Datenbanken. Einige dieser Templates sind sogar für den produktiven Einsatz supportet, sofern man gewisse Bedingungen berücksichtigt.

Erstellung eigener Templates

Selbst ein Template zu erzeugen ist auch nicht allzu schwierig. Zuerst ist die VM zu definieren, entweder im Web-GUI oder über das OVM CommandLine Interface (ovmcli). Dieses erreicht man mit SSH (Windows: Putty) auf Port 10000: `ssh -p 10000 admin@ovmmanager`

```
create Vm name=slot011 \
repository=slotreposan01 \
domainType=XEN_PVM \
osType=OL_6 \
bootOrder=DISK \
cpuCount=2 \
memory=2048 \
highAvailability=yes \
on ServerPool name=slot

create vnic macAddress=00:21:f6:01:00:aa \
   name=00:21:f6:01:00:aa network=vlanpublic on vm name=vm001

create VirtualDisk size=3.3 shareable=yes sparse=yes name=vm001_xvdd \
                   on repository name=myRepo

create VmDiskMapping slot=0  storageDevice=vm001_xvdd name=vm001_xvdd \
                     on Vm name=vm001
```

Nach der Erstellung kann die VM gestartet werden und das Betriebssystem wird installiert. Kleiner Tipp: für die Installation empfehle ich domainType=XEN_HVM, und die Anpassung auf XEN_PVM nach der Installation. Dies hat den Vorteil, dass man die Installation mit einem ISO-Image machen kann (bei Paravirtualisierung nur Netzwerkinstallation möglich).

Nach der Betriebssystem-Installation kann zusätzliche Software, wie beispielsweise die Oracle Datenbanksoftware (ORACLE_HOME) installiert werden.

Zum Schluss müssen wir diese Installation noch als Template vorbereiten, also die Mechanismen hinzufügen, damit nach dem Klonen beim ersten Boot die VM neu konfiguriert wird.

Cloning

Nachdem wir eine Master-VM oder OVM-Template erstellt haben, können wir dieses klonen und starten:

```
clone vm name=myTemplate destType=Vm destName=myVM \
         serverPool=myPool targetRepository=myRepo
start vm name=myVM
```

Im Runlevel 3 wird nun ein Script gestartet, das die VM bei Bedarf neu konfiguriert.

Oracle bietet uns hier die „Oracle VM Guest Additions“. Anders als man es sich von anderen Virtualisierungslösungen gewohnt ist, bringen diese Guest Additions nicht etwa optimierte Treiber für Disk, Netzwerk oder Bildschirm mit (das ist bei Linux schon im Kernel mit dabei). Es handelt sich um

eine Kommunikations- und Event-Handling Schnittstelle. Beispielsweise kann damit die VM ihre IP-Adresse an den OVM Manager melden.

Innerhalb der VM läuft ein Daemon "ovmd", welcher Meldungen von Aussen empfangen kann, und zwar nicht über Netzwerk, sondern direkt über einen Kommunikationskanal vom Host. Diese Guest-Additions sind auch in der Lage, Aktionen mit diesen Meldungen auszuführen, beispielsweise die Re-Konfiguration der VM.

Installation

Die Guest-Additions werden als RPM Paket geliefert. Bei Templates von Oracle sind sie bereits installert. Bei einer eigenen Installation eines Enterprise Linux ab DVD muss man zuerst noch das YUM Repository aktualisieren (wget http://public-yum.oracle.com/public-yum-ol6.repo) und ol6_addons (bzw. ol5_addons) aktivieren.

Danach kann die Software installiert und gestartet werden:

```
yum install libovmapi xenstoreprovider python-simplejson \
            ovmd ovm-template-config*
chkconfig ovmd on
/etc/init.d/ovmd start
```

Konfiguration

Für die Konfiguration wird ein Master-Konfigurationsscript "ovm-template-config" verwendet. Die Modul-Scripts befinden sich in /etc/templated/scripts. Geliefert werden die Module authentication , datetime, firewall, network, selinux, ssh, system, user. Der Mechanismus ist vergleichbar mit den System V Runlevel-Scripts. Mit nachfolgendem Befehl kann man sich alle Module anzeigen lassen:

```
vm001:~ #  ovm-chkconfig --list
name                configure   unconfigure reconfigure cleanup     suspend     resume      migrate     shutdown
authentication      on:90       off         off         on:10       off         off         off         off
datetime            on:50       off         off         on:50       off         off         off         off
firewall            on:41       off         off         off         off         off         off         off
network             on:50       off         off         on:50       off         off         off         off
selinux             on:30       off         off         off         off         off         off         off
ssh                 on:70       off         off         on:30       off         off         off         off
system              on:60       off         off         on:60       off         off         off         off
user                on:60       off         off         on:40       off         off         off         off
```

Mit diesen Modul-Scripts kann man die initiale Konfiguration vornehmen. Welche Parameter ein Script erwartet, lässt sich mit folgendem Befehl anzeigen.

```
ovm-template-config --human-readable --enumerate configure --script selinux
[('30',
  'selinux',
  [{u'description': u'SELinux mode: enforcing, permissive or disabled.',
    u'hidden': True,
    u'key': u'com.oracle.linux.selinux.mode'}])]
```

Unter "human-readable" verstehe ich zwar etwas anderes, aber man kann sich die Info wenigstens zusammensuchen.

Die Scripts lassen sich auch manuell ausführen:

```
ovm-template-config --console-input configure
```

Die Werte können statt auf der Konsole auch von Aussen als VM-Messages übergeben werden, oder innerhalb der VM via ovmd:

```
OVM> sendVmMessage Vm name=slot001 key=com.oracle.linux.network.device.0 \
     message=eth0 log=no
# ovmd -p com.oracle.linux.network.device.0=eth0
```

Eine weitere Möglichkeit ist die Nutzung von stdin:

```
ovm-template-config --stdin configure <<EOD
{"com.oracle.linux.selinux.mode": "disabled"}
{"com.oracle.linux.root-password": "ovsroot"}
EOD
```

Wichtig: die Zeile mit dem root-password muss als Letzte übergeben werden. Sobald diese Zeile eintrifft, werden die Scripts aufgerufen. Diese Anforderung gilt auch, wenn man die Werte von Aussen über das Messaging-System übergibt.

Wenn alles korrekt konfiguriert ist, muss nun das Template aktiviert werden, damit es beim nächsten Booten sich konfiguriert:

```
ovmd -s cleanup
service ovmd enable-initial-config
```

Erstellung eigener Script-Module

Dieses Konfigurationsframework von Oracle bietet uns auch die Möglichkeit, eigene Script-Module einzubinden. Beispielsweise zur Installation und Konfiguration von Oracle Software wie Grid-Infrastructure und ASM.

Erwartet wird ein Python Modul-Script. Da ich mit dieser Sprache nichts anfangen kann, habe ich nur ein minimales Python Wrapper-Script erstellt, das dann meine Shell- oder Perl-Scripts aufruft.

Wie bei Runlevel-Scripts besteht es aus einem Header, der die Einbindung in die Aufrufmechanismen regelt, und einem Body:

```
#!/usr/bin/env python
### BEGIN PLUGIN INFO
# name: myscript
# configure: 90
# description: Python wrapper Script to call myscript.ksh.
### END PLUGIN INFO
try:
    import json
except ImportError:
    import simplejson as json
from templateconfig.cli import main
from templateconfig.common import run_cmd
def do_enumerate(target):
    param = []
    if target == 'configure':
        param += []
    return json.dumps(param)
def do_configure(param):
    param = json.loads(param)
    run_cmd('/path/myscript.ksh')
    return json.dumps(param)
if __name__ == '__main__':
    main(do_enumerate, {'configure': do_configure})
```

Die eigentliche Arbeit macht also das myscript.ksh; Auch an dieses Script können Werte übergeben werden, welche dann mit `VARIABLE1=`ovmd -g variable1`` gelesen werden können.

Unser fertiges Script muss dann noch in die Aufrufmechanismen integriert werden:

```
ovm-chkconfig --add myscript
```

Das Script kann einzeln getestet werden. Zu beachten ist aber, dass hierbei die VM schon vollständig gestartet ist. Wenn das Script dann im Rahmen der initialen Konfiguration läuft, sind verschiedene Dienste noch gar nicht gestartet (z.B. NFS-Mounts).

```
ovmd -p key=value
ovm-template-config --script myscript --console-input configure
```

Beispiel

Im Vortrag wird dann gezeigt, wie man automatisiert per Script eine oder mehrere VM's durch Cloning erstellt und dann gleich die Grid Infrastructure für ein Standalone System oder für einen Cluster installiert und konfiguriert.

Kontaktadresse:
Martin Bracher
Trivadis AG
Europa-Strasse 5
CH-8152 Glattbrugg

Telefon:	+41 58 459 56 56
Fax:	+41 58 459 56 66
E-Mail	martin. bracher@trivadis.com
Internet:	www.trivadis.com