Wie kommt die Intelligenz in mein Oracle VM Template?

Manuel Hoßfeld
Oracle Deutschland
Geschäftsstelle Frankfurt

Schlüsselworte
Oracle VM, OVM, Template, Assembly, Automatisierung, Deployment

Einleitung
Bereits seit einiger Zeit gibt es mit Oracle VM Templates und Assemblies die Möglichkeit, "fertige" virtuelle Maschinen - z.B. mit Linux und einer vorkonfigurierten 11g Datenbank - auszurollen und diese direkt nach dem Start automatisch zu individualisieren, also z.B. mit eigenem Namen und IP-Adresse auszustatten.
Doch wie funktioniert dieser Mechanismus genau, und wie kann man ihn sich selbst zunutze machen? Dieser Vortrag beleuchtet darüber hinaus auch die Unterschiede zwischen Assemblies und Templates die sich in Zusammenhang mit dem automatisierten Deployment ergeben. Spätestens dann wenn der Wunsch aufkommt eigene Templates oder Assemblies mit der entsprechenden „Intelligenz" auszustatten stößt man auf einige nicht unbedingt offensichtliche Feinheiten, welche im Folgenden ebenfalls adressiert werden.

Oracle VM Templates und Assemblies – Eine Orientierung
Virtuelle Maschinen nicht „from scratch" aufbauen zu müssen, also konventionell durch manuelle Installationen von Betriebssystem und Applikationen, ist einer der großen Vorteile der Verwendung von Virtualisierungslösungen und „Infrastructure-as-a-Service"-Umgebungen. Stattdessen wird in diesen bevorzugt mit sog. Templates und Assemblies gearbeitet, die ein viel schnelleres und flexibleres Ausrollen von „fertigen" VMs ermöglichen. Da bedauerlicherweise selbst innerhalb der „Oracle-Welt" Begriffe wie „VM Template", „Assembly", „virtuelle Appliance" etc. nicht einheitlich verwendet werden und je nach Kontext unterschiedliche Fähigkeiten aufweisen, versucht dieser erste Abschnitt zunächst eine grundlegende Erklärung und Abgrenzung.
Hinweis: Wenn im Folgenden von „Oracle VM" (oder kurz: OVM) und dessen Möglichkeiten für Templates und Assemblies die Rede sein wird, ist ausschließlich Oracle VM Server (x86) in der Version 3.x gemeint. Die anderen Mitglieder der „Oracle VM Familie" besitzen zwar teilweise ähnliche Features (z.B. die sog. „VM Appliances" von Oracle VM VirtualBox) – diese werden hier allerdings nicht behandelt.

Aufbau / Unterschied von Templates und Assemblies

Technisch gesehen ist ein **Template** in Oracle VM lediglich eine „eingefrorene" VM, deren Bestandteile – im Mindestfall eine virtuelle Platte sowie die Konfigurationsdatei der VM – zu einem Archiv (i.d.R. als .tgz-Datei) zusammengeschnürt wurden. Dieses Archiv lässt sich nachdem es über den Oracle VM Manager in das Oracle VM Repository importiert zu einem späteren Zeitpunkt wieder „auftauen", d.h. es wird eine neue, lauffähige VM auf Basis des Templates erzeugt, welches nahezu vollständig identisch zu diesem ist. Die einzigen Unterschiede der neuen VM zum Template (bzw. zur „ursprünglichen VM" aus dem dieses erzeugt wurde) bestehen darin, dass Oracle VM einen anderen Namen und eine andere ID verwendet. Außerdem wird für die im Template definierten Netzwerk-

Interfaces jeweils natürlich eine neue MAC-Adresse vergeben, da es sonst zu Netzwerk-Problemen kommen würde.

An dieser Stelle ist es elementar zu verstehen, dass beim Erstellen einer VM aus einem Template über die o.g. „*äußeren*“ Anpassungen der VM selbst hinaus, keinerlei Änderungen „*innerhalb*“ dieser vorgenommen werden. Das bedeutet im Umkehrschluss dass z.B. der *Inhalt* der virtuellen Platte welches das Boot-Device abbildet absolut identisch ist mit dem, den das Template ursprünglich hatte. Konkret bedeutet dies dass die dann erstmals neu gestartete VM zwar ein „neues“ Netzwerk-Device (bzw. genauer gesagt eben nur eine neue MAC-Adresse) hat, aber ansonsten identisch wie ihr „Vorgänger“ konfiguriert ist. Sofern nicht DHCP verwendet wird, bekommt die VM also weder eine neue IP noch einen neuen Namen – was in der Regel nicht dem entspricht, was man erwarten bzw. sich wünschen würde. (Wie dennoch eine sinnvolle Neukonfiguration bzgl. Netzwerk etc. bei einer aus einem Template erzeugten VM erfolgen kann, wird an späterer Stelle noch erläutert werden.)

VM Templates sind also im Grunde genommen relativ „dumme“ Kopien der VMs aus denen sie erzeugt wurden. Wahrscheinlich ist dies auch einer der Gründe, warum seit OVM 3.x in Zusammenhang mit Templates im OVM Manager immer nur von „Klonen“ die Rede ist. (Sprich: Man erzeugt letztlich eine Kopie – entweder in Form einer lauffähigen VM oder eben wiederum als Template. Die Dokumentation [1] erläutert dies im Detail.)
Neben den gerade erwähnten prinzipiellen Einschränkungen „wissen“ Templates auch nichts über sich selbst oder ihre Umwelt, wie z.B. das Zusammenspiel mit anderen Servern / anderen VMs. Aus diesem Grund existiert mit den sog. **Assemblies** ein weiteres Konstrukt innerhalb einer OVM-basierten Infrastruktur:

Technisch gesehen handelt es sich bei einer **Assembly** zunächst um einen Satz aus einem oder mehreren VM Templates plus zusätzlichen Metadaten. Diese Metadaten beschreiben nicht nur die Templates und deren Konfiguration, sondern auch den Zusammenhang bzw. die Abhängigkeiten der zu erzeugenden VMs untereinander. Ein einfaches Beispiel wäre eine Assembly bestehend aus jeweils einer Datenbank-VM und einer damit verbunden Middleware-VM (bzw. genauer gesagt den dafür erforderlichen Templates). Beim Erzeugen und Starten der Middleware-VM würde diese dann (dank der Assembly-Metadaten) „wissen“, dass sie den Listener der Datenbank-VM auf Port 1521 findet und dass die DB natürlich vor dem Application Server gestartet sein muss damit überhaupt eine Verbindung zustande kommt.

Das Format dieser in Form einer XML-Datei abgelegten Assembly-Metadaten ist als offener Standard in Form des sog. „**Open Virtualization Formats“ (OVF)** definiert [2]. In der Regel werden sämtliche Bestandteile des Assemblies – ähnlich wie bereits bei einzelnen Templates – in ein Archiv gepackt. Man erhält dann eine .**OVA**-Datei (Open Virtualization Format Archive.)
Wohlbemerkt sagt der OVF-Standard nichts zur Semantik der abgelegten Metadaten aus – es handelt sich also eher um einen universellen „Container“ für VMs bzw. Templates, der keinen Anspruch erhebt zwischen verschiedenen Virtualisierungslösungen austauschbar und funktional identisch zu sein.

Automatisierung von Templates

Wie eingangs erwähnt passiert beim Erzeugen einer VM aus einem Templates nichts „Magisches“ – ohne entsprechende Anpassungen innerhalb des Betriebsystems wird die neue VM also z.B. keine vom Administrator gewünschte (oder aus einer vorher festgelegten Liste verfügbare) IP-Adresse erhalten. Auch die Konfiguration einer etwaigen enthaltenen Oracle-Datenbank wird genau identisch zu der ursprünglich im Template verwendeten sein.

Doch wie kann nun dennoch – automatisch oder zumindest halbautomatisch – eine sinnvolle Konfiguration beim erstmaligen Starten einer VM erfolgen, welche zuvor von einem Template erzeugt wurde?

Von Hand („alte Methode“)

Die ersten verfügbaren VM Templates hatten für die oben erwähnte initiale Konfiguration bzw. Individualisierung einer frisch daraus erzeugten VM eine jeweils eigene „First-Boot“-Mechanik implementiert. Vereinfacht gesagt handelte es sich um ein Boot-Skript, welches nach Prüfung einer Variable („Wird zum ersten Mal gebootet?“) eine Interview-Phase eingeleitet hat. D.h. es wurde eine Serie von Prompts ausgegeben, welche der Reihe nach Dinge wie Hostname, IP-Adresse, Gateway etc. abgefragt hat. Mit diesen Werten wurden anschließend die eigentlichen Konfigurationsdateien bestückt und die genannte Variable auf „false“ gesetzt, um den beschriebenen Vorgang bei zukünftigen Bootvorgängen nicht nochmal durchzuführen.

Diese Methode funktioniert zwar grundsätzlich, hat aber auch ein paar Nachteile:

- Ein Administrator muss beim ersten Boot der VM anwesend sein / eine entsprechende Konsole offen haben. (Zwar existiert die Möglichkeit des Fallbacks auf Default-Werte nach einem Timeout, aber die gewünschten Parameter kommen auf diese Weise natürlich nicht in die VM.) Eine Steuerung „von außen“ – etwa durch Angabe einer Parameterdatei o.ä. – ist also schlichtweg nicht möglich
- Auch fehlt ein nativer „Rückkanal“, über den eine laufende VM z.B. eine automatisch gesetzte IP-Adresse an die „Außenwelt“ (also den OVM Manager) melden könnte.
- Dadurch dass letztlich jeder Template-Autor selbst für den Aufbau und die konkrete Funktion der genannten „First-Boot“-Skripte verantwortlich war, unterschieden sich diese teilweise beträchtlich. Zwar wurden verschiedene Basis-Skripte meist wiederverwendet, aber letztlich fehlte eine komfortable und standardisierte Lösung.

Nutzung und Funktionsweise der VM Guest API

Die eben monierte Standardlösung um die Herausforderung der initialen Konfiguration von VMs zu meistern existiert inzwischen (seit ca. OVM 3.2) in Form der sog. „**VM Guest API**“.

Die VM Guest API ist eine bidirektionale Schnittstelle, mit deren Hilfe Nachrichten in Form von Key/Value-Paaren an eine laufende VM geschickt werden bzw. von dieser gelesen werden können. Die Kommunikation erfolgt dabei nicht etwa direkt mit den VMs, sondern immer über den Oracle VM Manager.

Aus technischer Sicht besteht die VM Guest API aus folgenden Bestandteilen:

- **ovmd**: Hierbei handelt es sich um einen kleinen Hintergrundprozess („daemon“), der innerhalb der entsprechenden VM läuft. Dieser implementiert bzw. „bedient“ gewissermaßen den oben beschriebenen bidirektionalen Kanal. Neben seiner Funktion als Daemon kann er auch direkt aufgerufen werden um z.B. aus einer VM eine Message an den OVM Manager zu schicken.
- **ovm-template-config**: Ein Satz von in der VM installierten, standardisierten und vorgefertigten Skripten, welche den ovmd verwenden, um eine initiale Konfiguration einer aus einem Template erzeugten VM vorzunehmen. Aufbau und Funktionsweise dieser Skripte sind ähnlich wie die der o.g. „alten Methode“, ohne die genannten Nachteile.
- **Hinweis**: Zusammen mit diversen dazugehörigen low-level Unterstützungspaketen werden *ovmd* und *ovm-template-config* auch als „**Oracle VM Guest Additions**“ bezeichnet.

- Entsprechende Kommandos aus dem OVM Command Line Interface (**OVMCLI**), z.B.:
  ```
  OVM> sendVmMessage vm name=ol6u4vm1 key=com.oracle.linux.hostname
  message=ol6u4.test.com
  ```
- **ovm-message:** Mit diesem (optionalen) Tool aus der OVM-Utilities Reihe [3] können von einem beliebigen Rechner aus der Verbindung zum OVM Manager hat, kommandozeilenbasiert Messages an eine laufende VM gesendet bzw. von dieser „abgeholt" werden. Ein Beispiel für letzeres kann z.B. so aussehen:

  ```
  # ./ovm_vmmessage -u admin -p password -h managerhost -v MeineVM
    -q key1
  Oracle VM VM Message utility 0.5.2.
  Connected.
  VM : 'MeineVM' has status :  Running.
  Querying for key 'key1.
  Query successful.
  Query for Key : 'key1' returned value 'Hugo'.
  Key set 5 minutes ago.
  ```

Die Dokumentation über die Guest Additions (und somit indirekt auch über die Guest API selbst) ist im Oracle VM Utilities Guide zu finden. [4]

Messages über die Guest API lassen sich auch mit der GUI im OVM Manager einsehen sowie an VMs schicken.

Nur solche VMs aus Templates, welche die o.g. „Guest Additions" bereits beinhalten, melden auch automatisch ihre IP-Adresse an den OVM Manager zurück. Sollte das entsprechende Feld also leer sein, fehlt höchstwahrscheinlich noch das entsprechende Paket. (Bezugsquellen und Installationsanweisungen finden sich ebenfalls in der o.g. Dokumentation.)

Templates selbst bauen

Template-Erstellung mittels "…save as template"

Genaugenommen gibt es seit OVM 3.x kein „…save as template" mehr, da der entsprechende Vorgang wie eingangs erwähnt inzwischen als „Klonen… in ein Template" bezeichnet wird.
Das Resultat ist in jedem Fall eine „dumme" Kopie der jeweiligen VM. Sollte diese nicht bereits die oben erwähnten „Guest Additions" (also ovmd und die ovm-template-config Skripte) in sich tragen, wird eine daraus erzeugte VM beim ersten Bootvorgang auch keinerlei Anstalten machen, sich automatisch zu konfigurieren bzw. zu individualisieren.

Daher ist diese Methode auch nur in wenigen Fällen sinnvoll und sei daher hier nur der Vollständigkeit halber erwähnt.

Oracle Virtual Assembly Builder (OVAB)

Im Rahmen des Application Servers Oracle Weblogic (Enterprise Edition) existiert mit dem Oracle Virtual Assembly Builder ein Werkzeug, welches wie der Name schon vermuten last zum Erstellen von Assemblies verwendet werden kann. Das Bundling mit der Oracle Middleware ist übrigens rein lizenztechnischer Natur und bedeutet nicht etwa, dass Oracle Datenbank-Nutzer das Tool nicht nutzen können oder sollen.
Kunden die einen Support-Vertrag für Oracle VM haben, verfügen übrigens implizit ebenfalls über eine Lizenz für den OVAB und müssen daher – solange es nur um die Paketierung reiner Linux-Assemblies (ohne weitere Bestandteile darin) geht – keine Middleware-Lizenzen anschaffen.

Die Grundidee beim OVAB ist es, mittels einer sog. „Introspektion" aus einer bereits vorhandenen Umgebung (die „auf Physik" laufen oder bereits virtualisiert sein kann) eine Art „Blaupause" dieser Umgebung zwecks wiederholtem und konfigurierbaren Ausrollen derselben in einer OVM-virtualisierten Infrastruktur zu erstellen. Die Bereitstellung dieser „Blaupause" erfolgt in Form einer Assembly, welche dann eine oder mehrere VM-Templates inkl. der notwendigen Metadaten enthält. Im einfachsten Fall kann man den OVAB aber auch als eine Art P2V („physical-to-virtual") - bzw. V2V („virtual-to-virtual") – Werkzeug verwenden.

Entscheidend ist jedenfalls, dass jegliche mit dem OVAB erzeugten Assemblies bzw. die darin enthaltenen Templates die oben erwähnte VM Guest API verwenden und bereits entsprechende Skripte z.B. für das Setzen der IP-Adresse beinhalten. Selbst ohne Verwendung weitergehender Funktionen (Verbindung einzelner Komponenten/VMs untereinander etc.) kann es aus diesem Grund nützlich sein, den OVAB zu verwenden.

Eine genaue Beschreibung der Möglichkeiten und Funktionsweise des OVAB würde den Rahmen dieses Vortrags sprengen, so daß an dieser Stelle für weitere Informationen auf die entsprechende OTN-Seite [5] verwiesen wird.

Nutzung von vorgefertigten Templates und Assemblies

Die sog. "Software Delivery Cloud" von Oracle für alle Downloads rund um Oracle VM ist unter http://edelivery.oracle.com/oraclevm zu finden. Da in der entsprechenden Sektion „Oracle VM Templates" jedoch sowohl Templates als auch Assemblies zu finden sind, gilt es hier aufzupassen und einige Dinge zu beachten:

Templates von eDelivery

Auch heute noch sind viele „klassische“ OVM Templates im eigentlichen bzw. ursprünglichen Wortsinne zu finden: D.h. als .tgz-Dateien verpackte Templates, welche direkt in den OVM Manager importiert und dann verwendet werden können. I.d.R. verfügen diese auch über eine entsprechende „First-Boot“-Mechanik wie in der Einführung erwähnt. Die tatsächliche „Intelligenz“ dieser älteren Templates kann also von Fall zu Fall variieren und erlaubt leider in keinem Fall eine Konfiguration von außen mittels der OVM Guest API.

“Assemblies” von eDelivery

Die OVM Templates neueren Datums sind technisch gesehen eigentlich Assemblies. D.h. sie kommen als .ova-Dateien daher und müssen im OVM Manager nach dem Import zunächst in ein eigentliches Template umgewandelt (d.h. genauer: ausgepackt) werden. Erst dann sind sie einsatzfähig. Als Vorteil ist jedoch in jedem Fall zu nennen, dass diese Sorte von Templates bereits mit den „VM Guest Additions“ ausgestattet ist. Dinge wie externes Setzen und Auslesen der IP-Adresse sind also möglich. Außerdem können diese Templates (dann allerdings als Assemblies) in die Software Library von Oracle Enterprise Manager Cloud Control importiert werden um dort als Basis für IaaS (Infrastructure-as-a-Service) VMs zu dienen. (Siehe dazu auch weiter unten.)

„Echte“ Assemblies im Sinne des OVAB (also als mehrteilige/mehrschichtige Blaupause einer komplexen Gesamt-Applikation) sind derzeit noch nicht auf eDelivery zu finden. Es ist jedoch zu erwarten dass dies in Zukunft der Fall sein wird.

Templates und Assemblies im OVM Manager

Lediglich im Sinne eines in der Einführung bereits erwähnten „Containers“ betrachtet der **Oracle VM Manager** zunächst eine **importierte Assembly**: Damit diese überhaupt für den Anwender von Nutzen ist, müssen die darin enthaltenen Templates zunächst als solche ausgepackt und separat abgelegt werden. Die zusätzlichen Metadaten werden hierbei *nicht* ausgewertet – ganz im Gegensatz zu den in diesem Vortrag ebenfalls erwähnten Werkzeugen *Oracle Enterprise Manager Cloud Control* sowie *Oracle Virtual Assembly Builder*.

Templates und Assemblies in Oracle Enterprise Manager Cloud Control

Dem ersten Anschein nach kann Cloud Control sowohl OVM Templates als auch Assemblies dazu verwenden, automatisiert (z.B. im Rahmen von IaaS über ein Self-Service Portal) neue VMs zu erzeugen. Doch es gibt einen entscheidenden Unterschied:

Die „intelligenten Features“ um die es in diesem Vortrag geht – d.h. die sinnvolle, automatische aber individualisierte Konfiguration einer neu erzeugten VM – werden lediglich beim Deployment von Assemblies verwendet. Auch wenn beide Vorgänge sehr ähnlich aussehen, handelt es sich im Hintergrund dennoch um jeweils andere Workflows. Mit anderen Worten:

Ein Deployment von VMs aus VM Templates verwendet *keinerlei* Möglichkeiten der bidirektionalen Kommunikation, wie diese durch die o.g. VM Guest API eingeführt wurden. Im Umkehrschluss bedeutet dies nicht nur, auf die weitergehenden Konfigurationsmöglichkeiten von mehrschichtigen Assemblies zu verzichten. Auch scheinbar simple Dinge wie das Setzen bzw. Rückmelden einer IP-Adresse aus einem vordefinierten Nummern-Pool und das anschließende automatische Konfigurieren eine EM Agenten sind ausschließlich mit den „modernen“/neuen VM Templates möglich, welche sich technisch eigentlich bereits als Assemblies (also OVAs mit nur einer Komponte/einer Schicht) darstellen.

In jedem Fall spielt es für die korrekte Funktionsweise keine Rolle, ob die jeweilige „Assembly“ z.B. vom OVAB erzeugt und dann in die Software Library von Cloud Control importiert wurde, oder ob sie bereits von Oracle vorgefertigt über den Bereich „Self-Update“ zur Verfügung gestellt und heruntergeladen wurde.

---

[1] Oracle VM User’s Guide, Kapitel 7.9: “Cloning a Virtual Machine or Template” : http://docs.oracle.com/cd/E35328_01/E35332/html/vmusg-vm-clone.html

[2] Wikipedia-Eintrag zum OVF-Format: http://en.wikipedia.org/wiki/Open_Virtualization_Format

[3] Oracle VM Utilities Guide: “2.5 Using Oracle VM Virtual Machine Messaging” http://docs.oracle.com/cd/E35328_01/E35333/html/vmutl-using-vmmessage.html

[4] Oracle VM Utilities Guide, Kapitel 3: “Using the Oracle VM Guest Additions” http://docs.oracle.com/cd/E35328_01/E35333/html/vmutl-guestadd.html

[5] Informationen zum „Oracle Virtual Assembly Builder” (OVAB) : http://www.oracle.com/technetwork/middleware/ovab/overview/index.html

Kontaktadresse:

Manuel Hoßfeld
Oracle Deutschland B.V. & Co. KG
Robert-Bosch-Str. 5
D-63303 Dreieich
Telefon: +49 (0) 6103-397494
E-Mail Manuel.Hossfeld@oracle.com