Automatisieren des Oracle VM Server per REST-Interface

Robert Marz, its-people GmbH

Oracle VM Server ist eine der Basis-Technologien, auf denen die Oracle Cloud aufbaut. Sie ist kostenlos und Open Source. Wird sie in größerem Maßstab eingesetzt, ist eine Automatisierung der Abläufe Pflicht. Leider sind die Informationen darüber dünn gesät.

Der Oracle VM Server wird über den Oracle VM Manager verwaltet, eine J2EE-Anwendung, die in einem – zu diesem Zwecke kostenfreien – Oracle WebLogic Server läuft. Der wohl erste Kontakt eines jeden Anwenders mit dem Oracle VM Manager findet über die Web Console statt. Diese ist hervorragend geeignet, sich einen Überblick über den Zustand der verwalteten VM Server, virtuellen Maschinen und den Storage zu verschaffen.

Ist das GUI-Konzept erst einmal verstanden, findet man sich schnell zurecht. Manuelle Änderungen an einzelnen Objekten sind mit wenigen Mausklicks durchgeführt. Ähnliche Änderungen an vielen Objekten auszuführen, kann allerdings schnell in eine Klickorgie ausarten und zur echten Qual werden. Bis zum Release 3.1.1 war die Web Console die einzige unterstützte Möglichkeit, Oracle VM Server zu verwalten *(siehe Abbildung 1)*.

Die Kommandozeile

Oracle hat mit der Version 3.2 die Kommandozeilen-Schnittstelle „OVM CLI“ eingeführt. Sie ermöglicht es dem Benutzer, sich via „ssh“ direkt auf dem Oracle VM Manager einzuloggen und einzelne Befehle oder ganze Skripte per Konsole abzusetzen. Leider hat sich Oracle entschieden, einen eigenen Parser und eine eigene Syntax für das „OVM CLI“ zu entwickeln. Während die Syntax zwar ein paar Holprigkeiten hat, etwa bei der Festlegung, wann Objekttypen abgekürzt und wann sie ausgeschrieben werden, ist sie doch eingängig. Der Parser hingegen ist sehr gewöhnungsbedürftig: Er kontrolliert jedes Zeichen bereits bei der Eingabe und quittiert mit entsprechenden Fehlermeldungen. Das führt dazu, dass er interaktiv kaum zu benutzen ist, sondern nur per Skript angesteuert wird *(siehe Abbildung 2)*.

„OVM CLI“ ist gut beschrieben. Es hat sein eigenes ausführliches Handbuch in der Dokumentation erhalten und kann synchron oder asynchron arbeiten: Die Ausführung wartet entweder, bis die einzelnen Befehle, etwa das Kopieren einer

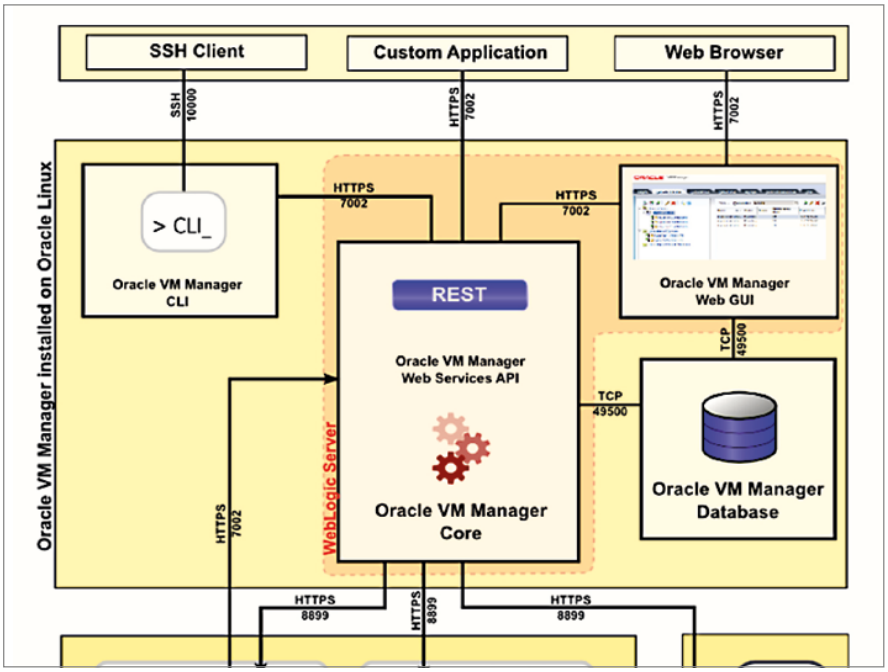

Abbildung 1: Die Architektur des Oracle-VM-Servers

Abbildung 2: Das Oracle VM Command Line Interface (CLI)

virtuellen Festplatte, abgeschlossen wurde, bevor sie mit dem nächsten Schritt fortfährt, oder sie liefert eine Job-ID zurück und fährt sofort mit dem nächsten Befehl fort. Für das Format, in dem die Antworten zurückgeliefert werden, kann zwischen lesbarem Text oder maschinenverarbeitbarem XML gewählt werden.

Das alles klingt eigentlich nach der idealen Lösung, wenn es um Automatisierung geht. Leider gibt es einen Haken: Die Authentifizierung ist zwar per „ssh"-Schlüssel möglich, aber erst, nachdem der Benutzer sich einmal nach jedem Neustart des OVM Manager per Kennwort angemeldet hat. Das bedeutet, automatisch gestartete Batch-Jobs, die sich kennwortlos per „ssh"-Schlüssel anmelden, funktionieren nicht mehr, wenn der VM Manager neu gestartet wurde.

Die VM-Rest-Schnittstelle

Wie auf dem Architekturbild gut zu sehen ist, nutzen sowohl die Web Console als auch die Kommandozeilen-Schnittstelle das VM-Manager-Web-Services-API, eine REST-Schnittstelle, die auch für eigene Anwendungen genutzt werden kann. Was liegt näher, als dieses API direkt zu nutzen? Schließlich bietet es ungefilterten Zugriff auf die komplette Funktionalität des VM Manager.

Referenz-Dokumentation mit Hindernissen

Vor der Benutzung gilt es allerdings, eine kleine Hürde zu überwinden: Es scheint keine Dokumentation zu geben. Das passende Handbuch „Oracle VM Web Services API Developers Guide for Release 3.4" ist erstaunlich dünn. Es enthält zwar ein Kapitel mit dem Namen „Using the API", in dem unter anderem erklärt wird, wie die Verbindung und Authentifizierung funktioniert, und es gibt auch Beispiele in Java und Python. Jedoch der wichtigsten Teil, nämlich eine Referenz der API-Funktionen, existiert nicht. Es gibt sie zwar, aber es braucht ein paar einfache Schritte:

- Das aktuelle Installationsmedium des OVM Manager, also das ISO-File, von eDelivery herunterladen
- Aus dem Wurzelverzeichnis die Datei namens „OvmSDK_3.4.2.1384.zip" (die Versionsnummer wird sich mit dem nächsten Release wieder ändern) auf den lokalen Rechner kopieren
- Entpacken

Danach finden sich im entpackten Verzeichnisbaum neben einer OVM-Web-Services-Client-Bibliothek (das ist ein „.jar"-File) Java-Beispiele, wie diese zu nutzen ist, sowie die „Full API Documentation" im HTML-Format. Warum Letztere nicht einfach im passenden Handbuch abgelegt ist, ist dem Autor unverständlich.

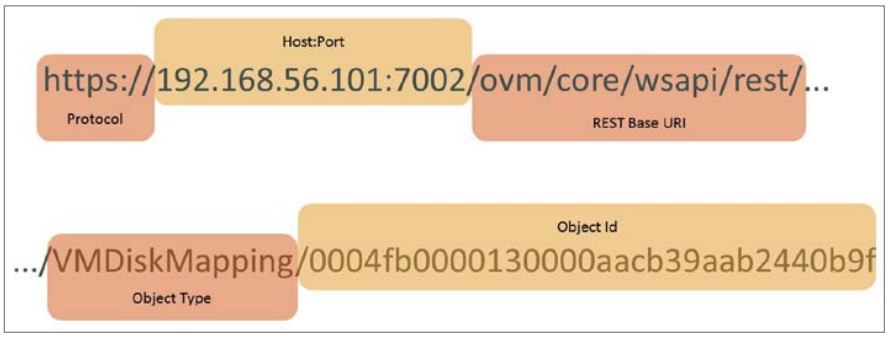

Abbildung 3: Eine typische OVM-REST-URL

HTTP Method	Operation	URI Path
GET	Get all	/ObjectType
	Get all IDs	/ObjectType/id
	Get by ID	/ObjectType/{id}
	Get child associations	/ObjectType/{id}/ChildType
	Get child association IDs	/ObjectType/{id}/ChildType/Id
PUT	Modify	/ObjectType/{id}
	Action	/ObjectType/{id}/action
	Add child association	/ObjectType/{id}/addChildType
	Remove child association	/ObjectType/{id}/removeChildType
POST	Create	/ObjectType
	Create child Object	/ParentType/{id}/ChildType
DELETE	Delete	/ObjectType/{id}
	Delete child Object	/ParentType/{parentId}/ChildType/{childId}

Abbildung 4: Übersicht über die http-Methoden und ihre Funktionen

Aufbau eines OVM-REST-Request

Jeder OVM-Rest-Request besteht aus den Komponenten URL, http-Methode, http-Header, Nutzlast und Antwort. Eine typische REST-URL sieht wie in *Abbildung 3* aus.

Diese Zeile könnte genauso auch in das Adressfeld eines Webbrowsers eingegeben werden. Aus Darstellungsgründen wurde die URL im Bild in zwei Zeilen umbrochen.

Die Komponenten aus der oberen Zeile (Protokoll, Host, Port und die REST Base URI) sind für jeden Aufruf gleich und werden nicht verändert. Der Teil danach bestimmt den Objekt-Typ und gegebenenfalls eine konkrete ID, mit dem interagiert werden soll. Was damit geschehen soll, bestimmt die http-Methode, mit der der Request an den Server gesendet wird. Mit einem Browser kann die Methode nicht gewählt werden. Alle URLs, die in die Adresszeile eingegeben sind, werden mit der Methode „GET“ abgesetzt.

Die Methoden sind wie Verben zu lesen. Das OVM-REST-API verwendet die Methode „GET“, um Informationen abzufragen; „PUT“ wird verwendet, um Objekte zu modifizieren oder Aktionen auszulösen. „POST“ erzeugt Objekte, während „DELETE“ diese wieder löscht. *Abbildung 4* zeigt eine Übersicht über die http-Methoden.

Jeder http-Request wird mit Kopf-Feldern an den Server gesendet. Das sind Variable, die der annehmende Server auswertet. Neben den klassischen Feldern wie „Authoriziation“ und „Cookie“ kennt das OVM-REST-API noch die Felder „Accept“ und „Content-Type“. Beide können jeweils „application/xml“ oder „application/json“

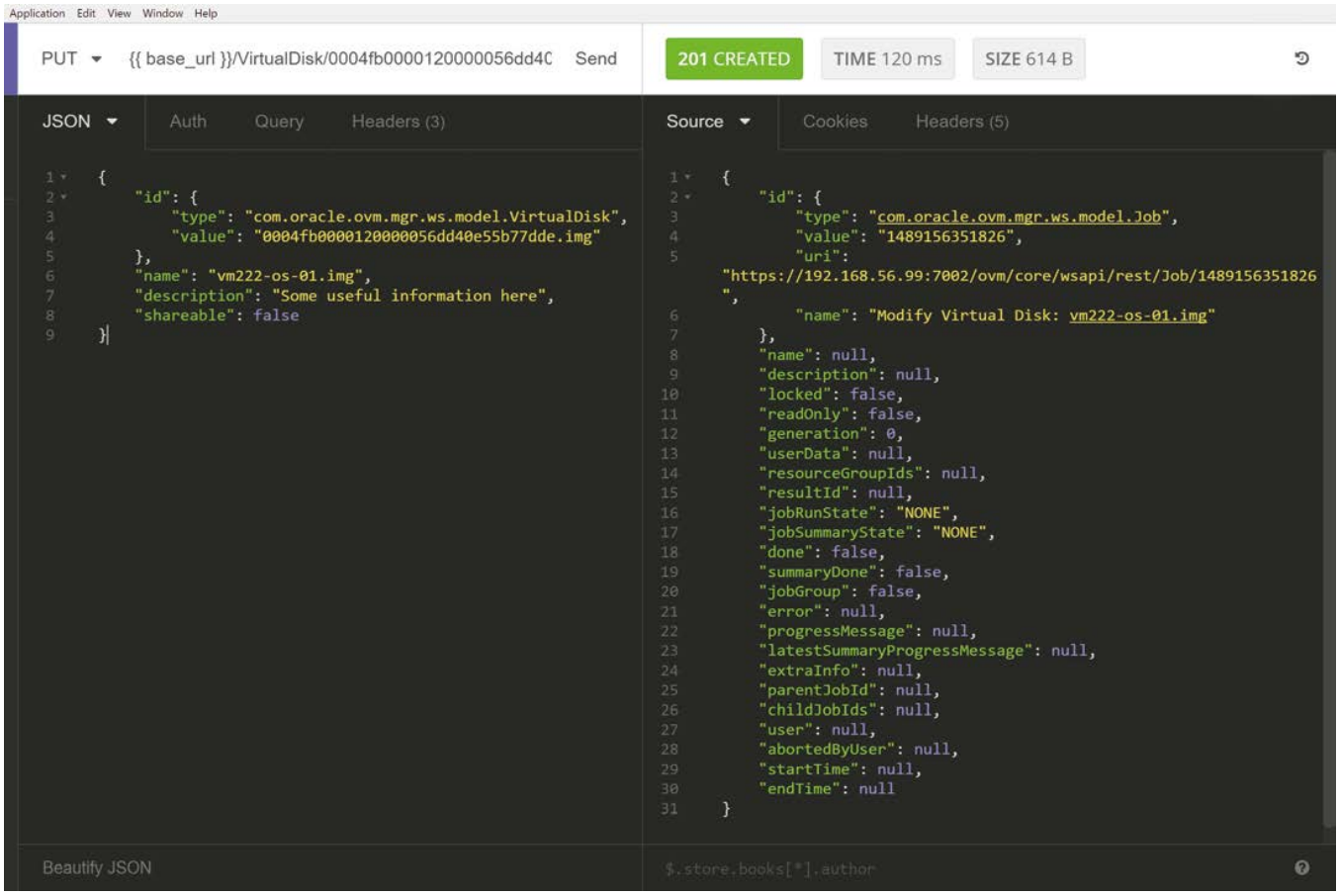

Abbildung 5: Ein Request mit Nutzlast und Antwort im JSON-Format

als Werte annehmen. Accept steuert dabei, in welchem der beiden Formate der Server antwortet, wohingegen „Content-Type“ bekannt gibt, wie die Nutzlast codiert ist.

Der Server antwortet in der Regel mit Job-IDs oder Objektbeschreibungen im gewünschten Format. Letztere eignen sich als Vorlage für die Nutzlasten, die zum Beispiel neue Objekte oder Änderungen beschreiben *(siehe Abbildung 5)*.

Authentifizierung

Das OVM-REST-API hat drei Wege, um sich gegenüber dem Server zu authentifizieren. Der einfachste ist, mit jedem Aufruf eine „http basic auth“ mitzusenden, also Benutzername und Kennwort codiert, aber nicht verschlüsselt in einem Header-Feld. Das ist sehr einfach zu implementieren. Der Server startet allerdings für jeden Request eine eigene Session, die erst nach einiger Zeit wieder beendet wird. Viele kurz aufeinanderfolgende Anfragen können deshalb zu einem erheblichen Ressourcenverbrauch auf der Serverseite führen.

Besser für den Server ist es, die REST-Login-Methode zu verwenden. Dazu wird als erste Anfrage einer Sitzung ein „POST“-Request zur URI „ovm/core/wsapi/rest/login“ mit „http basic auth“ geschickt. Der Server antwortet darauf mit einem Cookie namens „_WL_AUTHCOOKIE_JSESSIONID“. Dieses kann dann bei jedem weiteren Request der Sitzung als Anmelde-Information mitgeschickt werden. Damit kann der Server die Anfragen einer einzelnen Sitzung zuordnen und so schneller antworten. Da solche Sitzungen auch ablaufen können, muss jede Antwort überprüft und das Cookie gegebenenfalls erneuert werden.

Als dritter Weg sind SSL-Zertifikate vorgesehen. Nach dem initialen Aufsetzen ist das sehr bequem: Die Anmeldung erfolgt automatisch und es sind keine weiteren Schritte mehr nötig. Als Vorbereitung müssen die Zertifikate jedoch erzeugt und auf dem Server und jedem Client installiert werden.

Der Aufbau des OVM-REST-API

Das OVM-REST-API ist normalisiert. Jede Aktion, Funktion und Methode ist nur an einer Stelle abgelegt. Dieser Ort ist nicht immer intuitiv zu finden. Objekte vom Typ „VirtualDisk“ werden zum Beispiel mit „GET“ beziehungsweise „PUT /VirtualDiks/{virtualDiskId}“ abgefragt und verändert. Das Erzeugen oder Löschen sind hingegen Repository-Operationen: „POST /Repository/{repositoryId}/VirtualDisk” beziehungsweise „DELETE /Repository/{repositoryId}/VirtualDisk/{virtualDiskId}“. Werden Funktionalitäten, die es eigentlich geben müsste, vermisst, sind sie in der Regel doch vorhanden, nur an einem Ort, an dem man sie nicht vermuten würde.

Tipps und Tricks

Alle Aktionen sind asynchron: Es wird jeweils ein Job-Objekt zurückgeliefert, dessen Status abgefragt werden kann. Das REST-Interface ist für erfahrene Benutzer gedacht, die wissen, was sie tun. Virtuelle Medien, VMs oder ganze Repositories sind mit einem Aufruf ohne weitere Rückfrage gelöscht. Deswegen sollte Endanwendern kein direkter Zugang auf das API gewährt werden. Besser ist es, den Benutzern ein eigenes kleines API für spezifische Aufgaben, wie das Klonen von Festpatten oder Ähnlichem, zur Verfügung zu stellen. Beispiele dafür sind auf *„http://blog.its-people.de“* zu finden.

Robert Marz
robert.marz@its-people.de