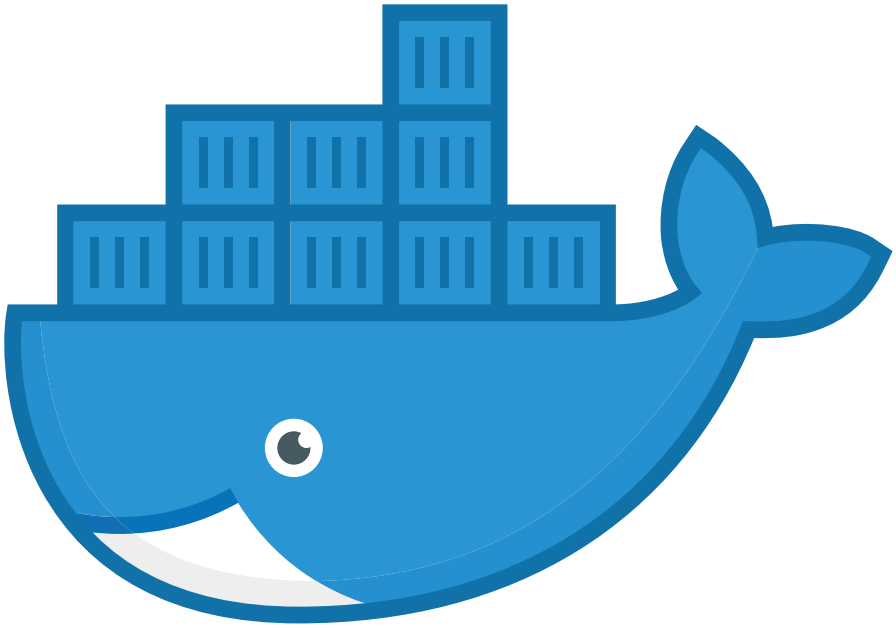

Oracle Unified Directory in Docker

Stefan Oehrli, Trivadis AG

Virtualisierung, Container beziehungsweise Docker sind seit Längerem aktuelle Themen der modernen IT. Viele Hersteller stellen entsprechend Docker-Images ihrer Produkte zur Verfügung. Oracle Unified Directory, ein schlanker, Java-basierter Verzeichnisserver, ist prädestiniert für den Betrieb in einer Container-Umgebung. Dieser Artikel zeigt, wie man Oracle Unified Directory in einem Docker-Image installiert, konfiguriert und anschließend als Container betreibt.

Für viele Oracle-Produkte wie Datenbanken, Fusion Middleware, Java etc. gibt es in der Oracle Container Registry *[1]* fertige Docker-Container, die sich zumindest für Entwicklungsarbeiten direkt nutzen lassen. Benötigt man eine speziellere Version oder möchte den Container an die eigenen Bedürfnisse anpassen, findet man im offiziellen Oracle Docker GitHub Repository *[2]* Docker-Files sowie Build- und Konfigurationsskripte – nur für Oracle Unified Directory (OUD) bietet Oracle aktuell keinen fertigen Docker-Container oder geeignete Build-Skripte an. Dies ist schade, zumal es den einen oder anderen Anwendungsfall gibt, in dem man einen LDAP-Server auch in einer Container-Umgebung nutzen möchte. Da Installation und Konfiguration von OUD keine Hexerei ist, steht dem Vorhaben „OUD Docker Image“ nichts im Weg.

Voraussetzungen

Oracle Unified Directory ist relativ genügsam, wenn es um die System-Anforderungen geht. Es sind lediglich ein entsprechendes Betriebssystem, ein aktuelles Oracle-JDK sowie rund 500 MB Platz für die Software und die OUD-Instanz erforderlich. Ein Betrieb von OUD ist auf jeder Umgebung möglich, in der ein Oracle-JDK installiert werden kann; also auch auf einem einfachen Raspberry Pi *[3]* mit Debian Linux, wobei es sich dabei nicht um eine zertifizierte Umgebung für OUD handelt. Informationen zu den zertifizierten Komponenten entnimmt man am besten der Zertifizierungsmatrix auf My Oracle Support *[4]*. Im Rahmen dieses Artikels werden wir uns auf folgende Versionen beschränken:

- Betriebssystem Oracle Enterprise Linux 7.5 (slim)
- Oracle Server JRE 8 Update 172
- Oracle Unified Directory 12.2.1.3.0
- Oracle Fusion Middleware Infrastructure 12.2.1.3.0
- Aktuelle Patch Set Updates vom April 2018
- OUD-Docker-Build-Skripte *[5]*
- OUD-Umgebungs-Skripte *[6]*

Weitere Versionen sind durch Oracle unterstützt. Im GitHub Repository des Autors *[5]* finden sich weitere Docker-Files für OUD 12c, OUD 11g, OUDSM 12c sowie für den etwas älteren Oracle Directory Server Enterprise Edition 11.1.1.7.0. Als Entwicklungsumgebung wird die Docker Community Edition 18.05.0 auf einem Laptop mit MacOS 10.13 verwendet. Es ist ohne Weiteres möglich, das vorgestellte Setup in aktuellen Docker-Umgebungen auf anderen Betriebssystemen nachzubauen.

Neben der lauffähigen Docker-Umgebung wird für die folgenden Arbeiten vorausgesetzt, dass sowohl die Docker-Build-Skripte als auch die Software lokal vorhanden sind. Die einfachste Methode dafür ist ein Git Clone vom GitHub Repository des Autors *[5]* zu erstellen. Die Links für die Downloads der oben aufgeführten Software sind, wie man es vom offiziellen Oracle Docker GitHub *[2]* Repository gewohnt ist, als „*.download"-Testdateien vorhanden.

OUD-Installationsmethoden

Mit dem neuesten Release von OUD hat Oracle verschiedene Varianten der Implementierung eingeführt.

- *Stand-alone OUD Server*
  Bei dieser Implementierungsmethode wird OUD als einfacher LDAP-Server mit geringem Platzbedarf eingesetzt. Die Administration erfolgt über die Kommandozeile („dsconfig", „ldapmodify" etc.) oder, wenn möglich, mit einem LDAP-Browser. Diese Methode wird für das OUD-Docker-Image verwendet.

- *Collocated OUD Server*
  Funktioniert mit OUD und Oracle Unified Directory Services Manager (OUDSM) in separaten Domains. Bei dieser Methode wird neben OUD zusätzlich Fusion-Middleware-Infrastruktur für den Oracle Unified Directory Services Manager (OUDSM) installiert. Diese Methode wird für das OUDSM-Docker-Image verwendet, wobei im OUDSM-Docker-Image lediglich die OUDSM-Komponente zum Einsatz kommt.

- *Collocated OUD Server*
  Funktioniert mit OUD und OUDSM in einer Domain. Bei dieser Methode wird neben OUD zusätzlich Fusion-Middleware-Infrastruktur für OUDSM installiert. OUD und OUDSM sind in einer WLS-Domain konfiguriert und werden gemeinsam administriert. Diese Methode ist komplexer, benötigt zusätzliche Installationsschritte und wird im Folgenden nicht weiter erläutert.

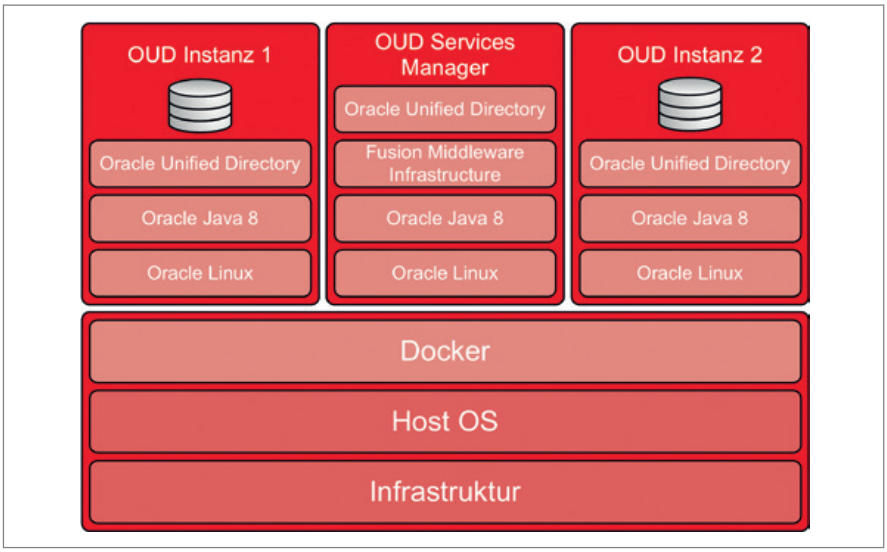

Abbildung 1: Docker-Image

Es ist möglich, sowohl OUD als auch OUDSM in einem Docker-Image zu installieren und zu nutzen. Doch auf diese Weise laufen mehrere Prozesse beziehungsweise Services in einem Container. Dies widerspricht dem Grundsatz, dass pro Container nur ein Service zu konfigurieren ist. Zudem lassen sich mit einem OUDSM mehrere OUD-Instanzen auf unterschiedlichen Hosts beziehungsweise Containern administrieren. Aus diesem Grund trennen wir die OUD- und OUDSM-Instanzen auf und erstellen zwei separate Docker-Images. *Abbildung 1* zeigt schematisch eine Docker-Umgebung mit mehreren OUD-Containern und einem OUDSM-Container.

Das Basis-Docker-Image

Bevor wir mit der eigentlichen Installation von OUD beginnen können, ist ein Docker-Basis-Image erforderlich. Im Docker-File von OUD und OUDSM wird mit „FROM oracle/serverjre:8" auf ein entsprechendes Basis-Image referenziert. In der Regel ist ein Image mit diesem Tag nicht vorhanden. Daher ist ein bestehendes Docker-Image, das Oracle Java 8 enthält, mit einem dazugehörigen Tag zu versehen.

Das Kommando „docker tag oracle/serverjre:1.8.0_172 oracle/serverjre:8" zeigt, wie ein bestehendes Docker-Image „oracle/serverjre:1.8.0_172" mit dem benötigten Tag versehen werden kann. Alternativ kann man das offizielle Oracle-Java-8-Image aus der Oracle Container Registry *[1]* laden. Nach dem Log-in kann das Image mit „docker pull" geladen und mit einem Tag versehen werden *(siehe Listing 1)*.

Als weitere Variante erstellt man ein eigenes Basis-Image mit Oracle Java 8.

```
docker login container-registry.oracle.com
docker pull container-registry.oracle.com/java/serverjre:8
docker tag container-registry.oracle.com/java/serverjre:8 \ oracle/serverjre:8
```

Listing 1

Im GitHub-Repository des Autors *[5]* sind ein entsprechendes Docker-File und eine kurze Beschreibung vorhanden. Mit dem Skripts „build.sh“ oder „docker build“ wird das gewünschte Image erstellt.

OUD-Image

Nachdem das Basis-Image vorhanden ist, muss die Software, also die OUD-Binaries„p26270957_122130_Generic.zip“ sowie der aktuellen PSU „p27742743_122130_Generic.zip“, in den Build-Ordner von OUD („$HOME/docker/OracleOUD/12.2.1.3.0“) kopiert werden. Anschließend lässt sich das OUD-Docker-Image mit den Kommandos „cd OracleOUD/12.2.1.3.0“ und „docker build -t oracle/oud:12.2.1.3.0“ erstellen. Zusammenfassend werden mit dem „docker build“-Kommando folgende Schritte aus dem Docker-File durchgeführt:

- Setzen der verschiedenen Umgebungsvariablen, abhängig von allfälligen Build-Argumenten für „ORACLE_ROOT“, „ORACLE_DATA“ oder „ORACLE_BASE“. Mit diesen Parametern lassen sich bei Bedarf die Verzeichnisstruktur der OUD-Installation, insbesondere das Verzeichnis für die OUD-Instanz, sowie das Oracle-Base anpassen. In der Regel ist dies aber nicht nötig.
- Erstellen des Benutzers und der Gruppen für die OUD-Installation sowie Anlegen der verschiedenen Verzeichnisse. Optional werden bei Bedarf noch fehlende Pakete aus dem Oracle Public YUM Repository nachinstalliert („tar“, „gzip“ und „libaio“).
- Kopieren der Docker-Hilfe-Skripte sowie, falls vorhanden, der Software-Pakete.
- Installation von OUD mit dem Hilfe-Skript „setup_oud.sh“. Es stellt sicher, dass die OUD-Software entpackt und mit einer Silent-Installation ein Standalone-OUD installiert wird. Im Anschluss an die Installation wird zusätzlich der aktuelle PSU installiert. Bei Bedarf kann das Skript die Software auch herunterladen. Dazu später mehr.
- Installation von OUD-Base mit dem Hilfe-Skript „setup_oudbase.sh“. Es lädt die aktuellste Version von OUD-Base direkt von GitHub.
- Festlegen der OUD-Ports, die später exportiert werden.
- Definition des Volume-Pfads für die persistenten Daten. Die OUD-Instanz sowie Log- und Konfigurationsdateien sind in einem separaten Verzeichnis „ORACLE_DATA“ beziehungsweise standardmäßig in „/u01“ abgelegt.
- Abschließend werden das Health-Check-Skript sowie das OUD-Start-Skript festgelegt.

Mit dem so erstellten Docker-Image lässt sich nun ein OUD-Container erstellen und starten. Das folgende Kommando *(siehe Listing 2)* startet im Hintergrund einen OUD-Container mit dem Namen „oudeng“ sowie dem Volume-Pfad „/Data/vm/docker/volumes/oudeng“.

Zusätzlich wird mit diesem Aufruf explizit das Port-Mapping angegeben. So

```
docker run --detach \
--volume /Data/vm/docker/volumes/oudeng:/u01 \
-p 1389:1389 -p 1636:1636 -p 4444:4444 \
-e OUD_INSTANCE=oud_test \
-e BASEDN=dc=example,dc=com \
--hostname oudeng --name oudeng \
oracle/oud:12.2.1.3.0
```

Listing 2

```
soe@gaia:~/ [ic12201] docker exec -it oudeng bash --login
Source environment for OUD Instance oud_docker
------------------------------------------------------------
 Instance Name       : oud_docker
 Instance Home (ok)  : /u01/instances/oud_docker
 Oracle Home         : /u00/app/oracle/product/fmw12.2.1.3.0
 Instance Status     : up
 LDAP Port           : 1389
 LDAPS Port          : 1636
 Admin Port          : 4444
 Replication Port    : 8989
------------------------------------------------------------
oracle@oudeng:/u00/app/oracle/ [oud_docker]
```

Abbildung 2: Interaktiver Zugriff auf „oudeng“ via „docker exec“

ist der Container von außen, also vom Host System, erreichbar. Auf diesen Container kann nun via Kommandozeile mit „docker exec“ interaktiv oder mit einem LDAP-Browser auf einen der exportierten Ports zugegriffen werden. *Abbildung 2* zeigt ein Beispiel, wie mit „docker exec“ eine „bash“-Shell im Container „oudeng“ gestartet wird. Auf diese Weise lässt sich die OUD-Instanz wie gewohnt mit der Kommandozeile verwalten.

Abbildung 3 zeigt den Directory Information Tree (DIT) der OUD-Instanz, angezeigt durch das Apache Directory Studio. Der Zugriff erfolgte via „localhost“ auf Port 1389 beziehungsweise mit der LDAP-URL „ldap://localhost:1389“.

Wie ersichtlich, enthält die Instanz bereits den Suffix „dc=postgasse,dc=org“ mit rund 100 Einträgen unter „ou=People, dc=postgasse,dc=org“. Dies liegt daran, dass durch das Startskript beim ersten Start des OUD-Containers automatisch eine Instanz angelegt wird. Dabei werden verschiedene Standartwerte verwendet, die sich durch das Setzen von Umgebungsvariablen steuern lassen. Dreh- und Angelpunkt ist das OUD-Startskript „start_oud_instance.sh“. Es prüft, ob eine OUD-Instanz vorhanden ist. Falls nicht, wird mit dem Skript „create_oud_instance.sh“ eine solche erstellt. Bei Bedarf werden dabei spezifisch Konfigurationsskripte ausgeführt, um die OUD-Instanz zu konfigurieren. *Abbildung 4* zeigt schematisch den Ablauf beim Start beziehungsweise Erstellen des OUD-Containers.

Neben „start_oud_instance.sh“ und „create_oud_instance.sh“ werden folgende Hilfe-Skripte im Docker-Container genutzt:

- „check_oud_instance.sh“ wird für den Health Check des Docker-Containers genutzt und prüft den Status der OUD-Instanz. Es ist verantwortlich für die Angabe eines Status „healthy/unhealthy/starting“ beim Aufruf von „docker p“.
- „config_oud_instance.sh“ prüft, ob entsprechende Konfigurationsdateien („*.sh“, „*.conf“ oder „*.ldif“) im Verzeichnis „${OUD_INSTANCE_ADMIN}/create“ vorhanden sind. Falls ja, werden diese sequenziell ausgeführt.
- „create_oud_instance.sh“ erstellt bei Bedarf eine OUD-Instanz. Basis-Instanz-Parameter lassen sich mit Umgebungsvariablen anpassen. Das Skript generiert explizit ein Passwort für den Root-Directory-Benutzer und legt dieses in der Datei „${OUD_INSTANCE_ADMIN}/etc/${OUD_INSTANCE}_pwd.txt“ ab.
- „setup_oud.sh“ wird nur beim Build des Docker-Image zur Installation von OUD benötigt.

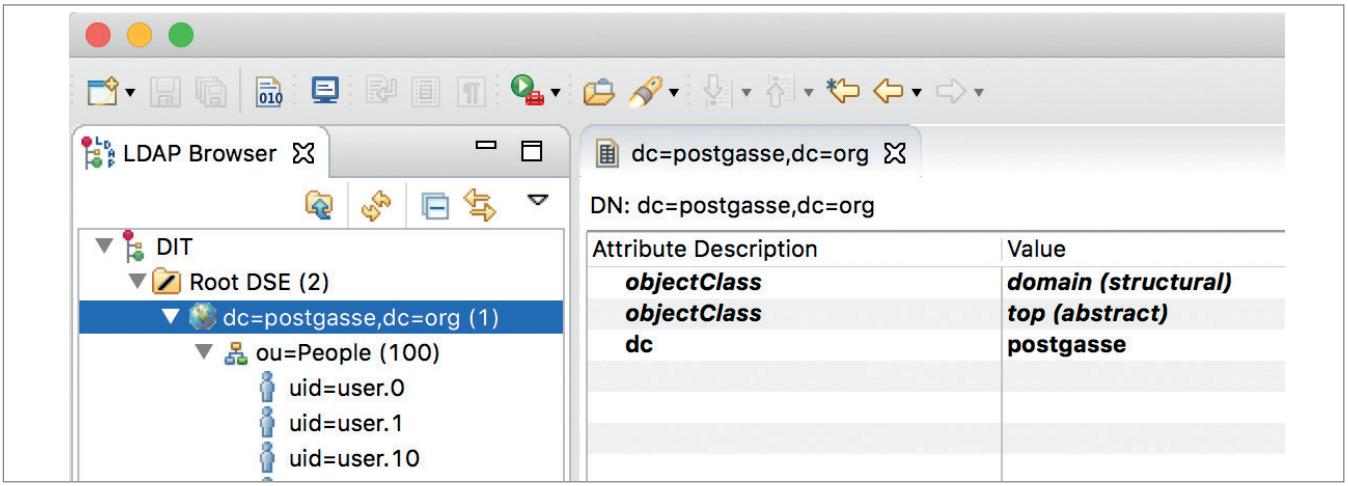

Abbildung 3: Zugriff mit Apache Directory Studio

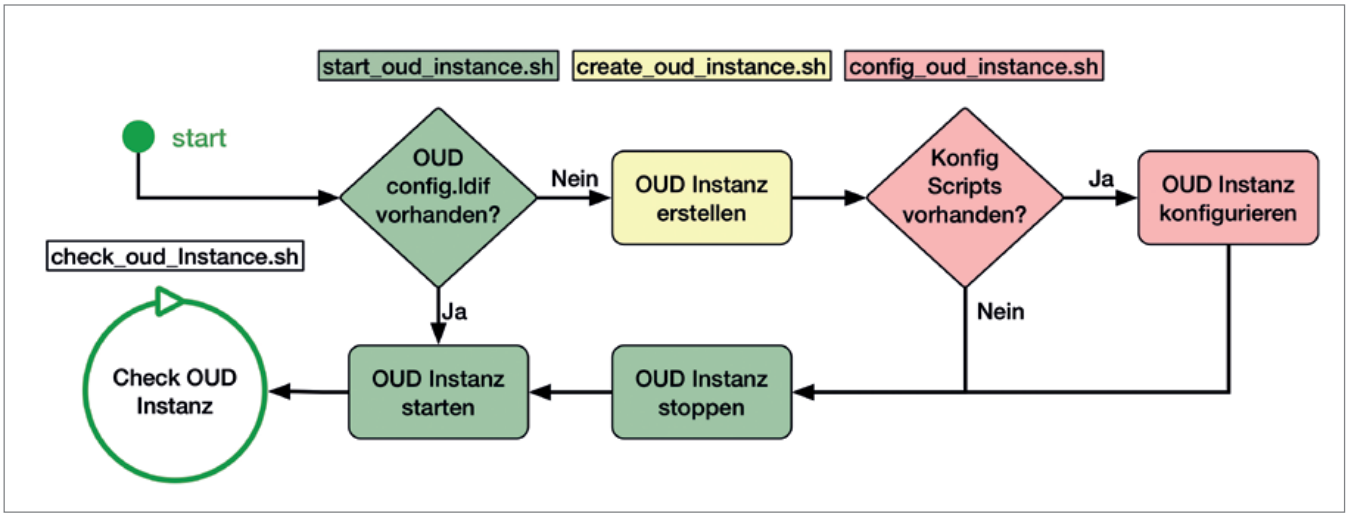

Abbildung 4: Ablauf beim Start der OUD-Instanz

- „setup_oudbase.sh“ wird beim Build des Docker-Image zur Installation der OUD-Base-Umgebungsskripte genutzt.
- „start_oud_instance.sh“ startet die OUD-Instanz und definiert Signal-Handler für „SIGINT, „SIGTERM“ und „SIGKILL“. Fehlt die OUD-Instanz, wird „create_oud_instance.sh“ aufgerufen.

Die Parametrisierung der Skripte erfolgt grundsätzlich via Umgebungsvariable bei der Instanziierung des Docker-Containers mit dem Parameter „-e“. Mit wenigen Ausnahmen ist eine nachträgliche Anpassung nicht sinnvoll. Ein Beispiel für diese Parametrisierung ist im folgenden Code-Beispiel ersichtlich. Dieser Aufruf von „docker run“ setzt explizit einen OUD-Instanz-Namen und einen alternativen BaseDN *(siehe Listing 3)*.

Neben „OUD_INSTANCE“ und „BASEDN“ können noch weitere Parameter zur Konfiguration des OUD-Containers beziehungsweise der OUD-Instanz genutzt werden. *Tabelle 1* gibt eine Übersicht.

Mit der Kombination der Umgebungsvariablen und den Konfigurationsskripten lassen sich verschiedene OUD-Instanzen für unterschiedliche Use-Cases anlegen. Fehlen die gewünschten Konfigurationsskripte, kann man mit „CREATE_INSTANCE=FALSE“ das automatische Erstellen der Instanz umgehen und manuell eine OUD-Instanz anlegen. Diese wird bei einem späteren Starten des Containers durch „start_oud_instance.sh“ automatisch gestartet. Läuft dabei etwas schief, hat man die Möglichkeit, direkt im Docker-Container die OUD-Log-Dateien zu prüfen oder mit „docker logs“ die OUD-Instanz-Log-Datei anzuzeigen. Mit „docker logs <CONTAINER>“ sieht man zudem immer auch den „STDOUT“ der Hilfe-Skripte beim Erstellen, Starten oder Stoppen einer OUD-Instanz.

Das OUDSM-Image

Der Aufbau des OUDSM-Image erfolgt analog zum OUD-Image. Neben den OUD-Binaries müssen zusätzlich noch Oracle Fusion Middleware Infrastructure „p26269885_122130_Generic.zip“ sowie das aktuelle WebLogic-PSU „p27342434_122130_Generic.zip“ in den Build-Ordner kopiert werden. Anschließend kann das OUDSM-Docker-Image mit den Kommandos „cd OracleOUDSM/12.2.1.3.0“ und „docker build -t oracle/oudsm:12.2.1.3.0“ erstellt werden.

Der Ablauf beim Build ist ähnlich wie beim OUD-Docker-Image. Der wesentliche Unterschied ist das Installationsskript „setup_oudsm.sh“, das beim OUDSM-Image eine „Collocated“-Installation von Fusion Middleware Infrastructure und Unified Directory erstellt. Zudem sind die oben beschriebenen Hilfe-Skripte für die Bedürfnisse von WebLogic beziehungsweise dem OUDSM angepasst und entsprechend umbenannt worden. Aus einem „start_oud_instance.sh“ wurde dem zufolge ein „start_oudsm_domain.sh“. Zusätzlich gibt es für das Erstellen der OUDSM-WebLogic-Domain das Python-Skript „create_oudsm_domain.py“. Das OUDSM-Image wird ausschließlich für den Betrieb eines Oracle Unified Directory Services Manager verwendet. Daher ist eine weitere Konfiguration grundsätzlich nicht nötig. Der gewünschte OUDSM-Container kann direkt mit dem Kommando in *Listing 4* erstellt werden.

Bei diesem Container wurde durch die Verwendung von „--volume“ ebenfalls explizit ein Volume angegeben. Die Zuweisung der Ports wurde mit dem Parameter „-P“ jedoch Docker überlassen. Mit dem Kommando „docker port“ können die verwendeten Ports nach dem Start des Containers angezeigt werden. Für den oben gestarteten Container wurden folgende Ports gewählt *(siehe Listing 5)*. *Abbildung 5* zeigt, wie man sich via „localhost“ auf den Port 32769 verbindet und mit OUDSM auf den zuvor erstellten OUD-Container „oudeng“ zugreift.

```
docker run --detach \
--volume /Data/vm/docker/volumes/oudeng:/u01 \
-p 1389:1389 -p 1636:1636 -p 4444:4444 \
--hostname oudeng --name oudeng \
oracle/oud:12.2.1.3.0
```

Listing 3

Variable	Verwendung
OUD_INSTANCE	OUD-Instanz-Name, Standardwert „oud_docker“
CREATE_INSTANCE	Flag zum Erstellen der OUD-Instanz beim Starten, Standardwert „TRUE“
OUD_INSTANCE_BASE	Basis-Verzeichnis für die OUD-Instanzen, Standardwert „$ORACLE_DATA/instances“
OUD_INSTANCE_HOME	OUD-Instanz-Home-Verzeichnis, Standardwert „${OUD_INSTANCE_BASE}/${OUD_INSTANCE}“
OUD_INSTANCE_ADMIN	OUD-Instanz-Admin-Verzeichnis, Standardwert „${ORACLE_DATA}/admin/${OUD_INSTANCE}“
ADMIN_USER	OUD-Directory-Root-Benutzer, Standardwert „cn=Directory Manager“
ADMIN_PASSWORD	Password für den OUD-Directory-Root-Benutzer, wird automatisch generiert und in „$PWD_FILE“ abgelegt. Durch Setzten von „ADMIN_PASSWORD“ kann explizit ein Passwort angegeben werden, das allerdings im Container als Umgebungsvariable gesetzt bleibt.
PWD_FILE	Password-File mit dem Clear Text Password vom Directory-Root-Benutzer, Standardwert „${OUD_INSTANCE_ADMIN}/etc/${OUD_INSTANCE}_pwd.txt}“
SAMPLE_DATA	Flag für das Erstellen der Beispieldaten, mögliche Werte sind „TRUE“, „FALSE“ oder Anzahl der Datensätze, Standardwert „TRUE“
OUD_INSTANCE_INIT	Verzeichnis für die Instanz-Konfigurationsdateien, Standardwert „$ORACLE_DATA/scripts“

Tabelle 1

```
docker run --detach \
--volume /Data/vm/docker/volumes/oudsm:/u01 \
-P --hostname oudsm --name oudsm \
oracle/oudsm:12.2.1.3.0
```

Listing 4

```
docker port oudsm
7002/tcp -> 0.0.0.0:32768
7001/tcp -> 0.0.0.0:32769
```

Listing 5

Die korrekte IP-Adresse des Containers „oudeng“ erhält man durch das Kommando „docker inspect <CONTAINER>“. Wenn man nicht alle Informationen zum Container anzeigen will, kann man dies, wie im Beispiel in *Listing 6*, direkt auf die IP-Adresse filtern.

OUD Base

Wie angemerkt, wird bei den Docker-Images für OUD und OUDSM sowie ODSEE immer OUD Base mit installiert. Es ist eine kleine Sammlung von Skripten, Umgebungsvariablen und Aliasen für die Kommandozeilen-Administration der Oracle Directory Server. Für den Betrieb der Docker-Container ist OUD Base nicht zwingend erforderlich. Gleichwohl erleichtert es einem die Arbeit auf der Kommandozeile.

Speziell wenn mehrere OUD-Instanzen konfiguriert wurden, liefern die Aliase „oud_status“ oder „oud_up“ immer Informationen wie Instanz-Name, Status, Oracle-Home, Ports etc. *Abbildung 2* weiter oben zeigt eine entsprechende Ausgabe von „oud_status“ beim Log-in. Weitere Informationen sowie Installationspakete findet man im OUD Base GitHub Repository des Autors *[5]*.

Die Größe der Docker-Images optimieren (für Fortgeschrittene)

Bei dem aufgezeigten Vorgehen für die Erstellung der Docker-Images für OUD und OUDSM wurde nicht speziell auf die Größe geachtet und die Software einfach im Build-Ordner abgelegt. Das „COPY“-Kommando im „Dockerfile“ stellt bei der Ausführung von „docker build“ sicher, dass die Software in das dazugehörige Image kopiert wird. Dies führt dazu, dass die Images für OUD und OUDSM beträchtlich größer werden als eigentlich nötig, was an der Art und Weise liegt, wie Docker die Images schichtweise aufbaut.

Ein Löschen auf einem nachfolgenden Layer bewirkt nur ein logisches Löschen, physisch bleibt die Software weiterhin Bestandteil des Layers. Eine einfache Lösung dieses Problems ist der Parameter „—squash“ bei neueren Docker-Versionen. Der Parameter wird zusätzlich bei „docker build“ angegeben und stellt sicher, dass alle neu erstellten Layer am Ende zusammengeführt werden. So können die oben aufgeführten „docker build“-Kommandos einfach mit „—squash“ ergänzt werden, um kleinere Images zu erhalten.

Alternativ dazu bietet es sich an, die Software beim Erstellen des Image direkt in einem „RUN“-Kommando herunterzuladen, auszupacken, zu installieren und am Schluss auch gleich zu löschen. Die verwendeten Setup-Skripte „setup_oud.sh“ und „setup_oudsm.sh“ sind entsprechend vorbereitet und können die Software nach dieser Methode installieren. Dazu muss beim Build mit dem Parameter „--add-host“ lediglich ein Host

```
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' oudeng
```

Listing 6

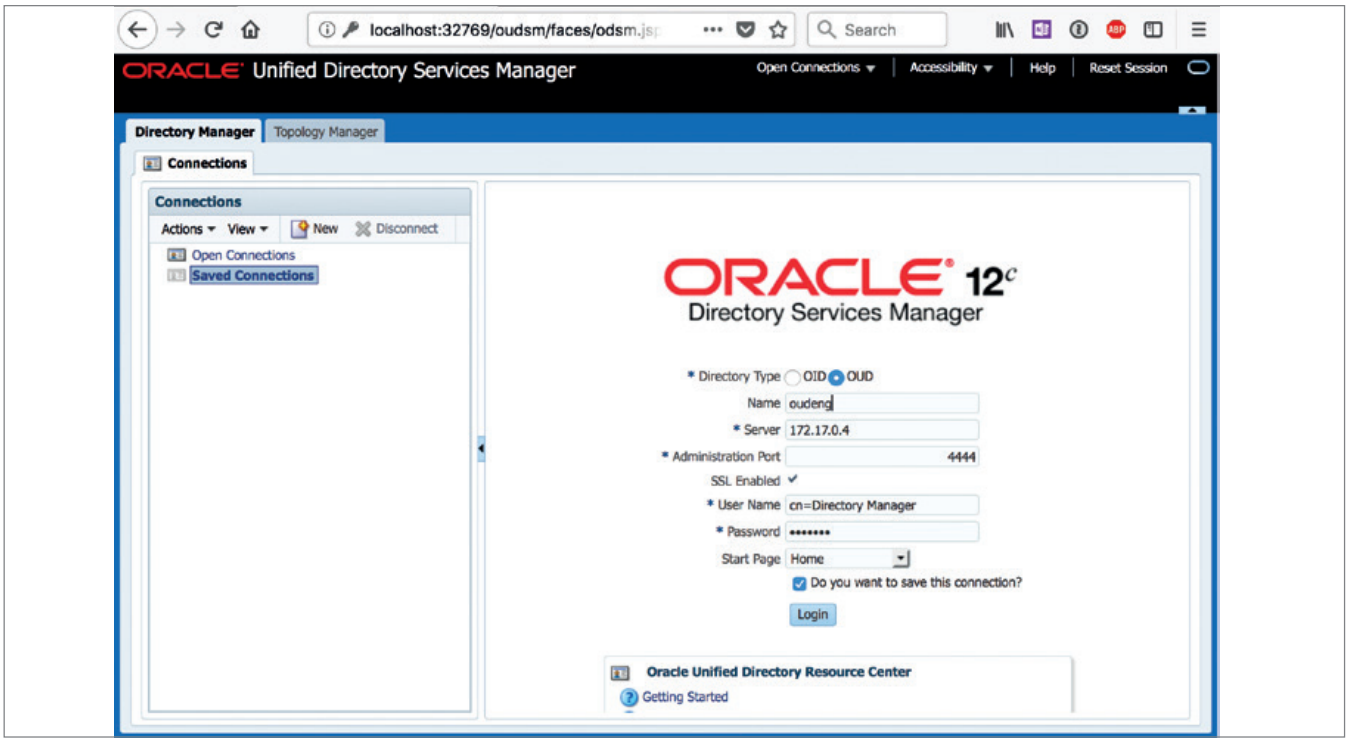

Abbildung 5: Log-in im Oracle Unified Directory Services Manager

```
cd OracleOUD/12.2.1.3.0
rm *.zip
docker build --add-host=orarepo:172.17.0.2 \
-t oracle/oud:12.2.1.3.0 .
```

Listing 7

beziehungsweise dessen IP-Adresse angegeben werden, von wo die Software heruntergeladen werden kann. Das Beispiel in *Listing 7* zeigt ein angepasstes Build-Kommando mit der Angabe einer IP-Adresse für den Host „orarepo".

Im Beispiel handelt es sich ebenfalls um einen Docker-Container mit einem Apache Webserver, der nichts anderes macht, als lokal via HTTP die Software zur Verfügung zu stellen. Mehr dazu im Blog-Post „Smaller Oracle Docker Images" *[7]*.

Fazit

Egal, ob man Oracle Unified Directory nur einmal anschauen möchte oder einen konkreten Anwendungsfall verfolgt – mit der hier vorgestellten Methode lässt sich Oracle Unified Directory schnell und einfach in einem Docker-Container aufsetzen und betreiben. Die Möglichkeit, dass die OUD-Instanzen manuell oder mit vordefinierten Skripts konfiguriert werden können, liefert die nötige Flexibilität, spezifische Bedürfnisse abzudecken.

Die verschiedenen Images für OUD, OUDSM und ODSEE wurden im Rahmen von Kunden-Projekten bereits mehrfach genutzt, um konkrete Problemstellungen zu lösen, etwa eine Ad-hoc-Test-Instanz für die Analyse von OUD-Problemen im Rahmen eines Service-Request, Test-Migration von ODSEE nach OUD, Entwicklung eines OUD-Proxy für die Integration weiterer Verzeichnisse, Einsatz von Oracle Enterprise User Security mit OUD und vieles mehr.

Mit einem „docker run" ist ein neuer OUD-Container schnell erstellt und man hat Zeit, sich auf das Wesentliche zu konzentrieren, ohne sich zuerst mit der Installation zu befassen. Zudem steht einer produktiven Nutzung von OUD auf Docker grundsätzlich nichts im Weg. Auch wenn Oracle bis heute noch keine offiziellen OUD-Docker-Images zur Verfügung stellt, wird gemäß Oracle-Support OUD auf Docker grundsätzlich unterstützt.

Quellen

[1] Oracle Container Registry: *https://container-registry.oracle.com*
[2] Oracle Docker GitHub: *https://github.com/oracle/docker-images*
[3] Blog Post OUD to go on Raspberry Pi Zero: *http://url.oradba.ch/2AHzfbi*
[4] My Oracle Support OUD 12c Zertifizierung: *https://url.oradba.ch/MOSoud12c*
[5] Stefan Oehrli GitHub Docker Repository: *https://github.com/oehrlis/docker*
[6] OUD-Umgebungs-Skripte: *https://github.com/oehrlis/oudbase*
[7] Blog Post Smaller Oracle Docker Images: *https://url.oradba.ch/2ut6jEH*

Stefan Oehrli
stefan.oehrli@trivadis.com