Docker Container für das Deployment eines WebLogic Clusters

Thomas Robert
ORACLE Deutschland B.V. & Co. KG
Geschäftsstelle Hamburg

Schlüsselworte

Docker, Swarm, Container, WebLogic, Cluster, Cloud, Oracle Container Cloud Service, OCCS

Einleitung

„*Entwickler können die für sie relevanten Oracle Technologien jetzt als Docker Container nutzen, um Cloud-Anwendungen zu entwickeln und einzusetzen.*“[1] Mit dieser Presse-Ankündigung vom 20. April 2017 gibt Oracle die Unterstützung für das Deployment von WebLogic basierten Anwendungen in Docker Containern bekannt. Container-Technologien vereinfachen und beschleunigen die Entwicklung und das Testen moderner Web-, Mobile- und IoT-Applikationen. Auch für den produktiven Betrieb ermöglichen Container vorhersagbare und reproduzierbare Deployments. Sie vereinfachen und beschleunigen DevOps Zyklen und verringern signifikant die Fehlermöglichkeiten, die durch unterschiedliche oder fehlerhafte Konfigurationen der Zielumgebung entstehen.

Grundlagen

Oracle WebLogic[2] ist einer der am weitesten verbreiteten Applikationsserver für den produktiven Betrieb von Java EE Anwendungen. Seine Multitenancy Option ermöglicht den Aufbau auch von Microservice-Architekturen. Vielfältige Hochverfügbarkeits-Funktionalitäten, die mehrere Rechenzentren umspannen können, erlauben einen hochperformanten, unterbrechungsfreien Betrieb komplexer Applikationen.

„***Docker*** *ist eine Open-Source-Software, die dazu verwendet werden kann, Anwendungen mithilfe von Betriebssystemvirtualisierung in Containern zu isolieren.*“[3] Im Gegensatz zu virtuellen Maschinen (VMs), in denen eine komplette Rechnerarchitektur (inklusive Betriebssystem) auf Basis eines Hypervisors abgebildet wird, enthalten Container nur die jeweils darin „verpackte“ Applikation mit den von ihr benötigten Libraries und Konfigurationsdateien. Docker basiert auf der Linux Container Architektur, deren Aufgabe es ist, Applikations-Ressourcen von ihrer Umgebung zu isolieren. Somit nutzen alle Container-Prozesse eines Rechners zwar den selben Betriebssystem-Kernel mit seinen Treibern. Dennoch sind sie von anderen (Container und nicht-Container) Prozessen derart isoliert, dass z.B. Prozesse und Dateisysteme außerhalb des eigenen Containers nicht sichtbar sind. Außerdem können die verfügbaren Rechnerressourcen (z.B. CPU, Hauptspeicher, Disk I/O) für jeden Container begrenzt werden.

Natürlich lassen sich Container auch innerhalb von virtuellen Maschinen betreiben. Dies ist in der Praxis sogar der Regelfall. Der Vorteil der Container liegt hier in der deutlich reduzierten Anzahl der benötigten VMs und in der feiner granularen Aufteilung der Systemressourcen.

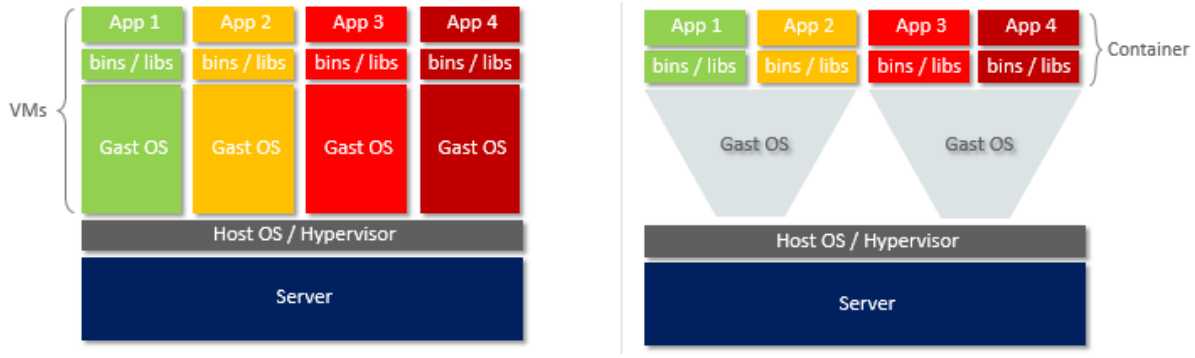

Abb. 1: VM vs. Container

Während Linux Container sich vor allem mit der Verwaltung von Prozessen und der Zuordnung von Computer-Ressourcen zu diesen Prozessen beschäftigen, vereinfacht Docker die Nutzung solcher Container durch eine übersichtliche Schnittstelle und ein umfangreiches Angebot an Management-Funktionen zur Erstellung und Verwaltung. Docker unterscheidet zwischen „Images“ (den „Blaupausen“ einer Applikation und deren Ressourcen) und „Containern“ (den Prozess-Instanziierungen eines Images). So können aus einem Image beliebig viele Container gestartet werden. Images lassen sich – im Gegensatz zu Containern – auf andere Rechnerumgebungen kopieren. „Docker Hub“ stellt eine öffentliche Online-Registry zur Verfügung, in der Docker Images abgelegt werden können um einen gemeinsamen Zugriff auf Applikationen zwischen Entwickler-Communities zu ermöglichen.

Clustering von Docker Containern

Die Isolation von Docker Containern umfasst natürlich auch die netzwerkseitige Abschirmung der in einem Container laufenden Prozesse. Jeder Container besitzt seinen eigenen virtualisierten Netzwerk-Stack, z.B. mit jeweils eigener IP Adresse. Hierdurch wird die Netzwerkkommunikation zwischen Containern standardmäßig verhindert. Innerhalb eines Rechners können Docker Container mit dem „--link“ Parameter netzwerkmäßig verbunden werden. Dieser Parameter ist jedoch inzwischen „deprecated“ (veraltet). Er funktioniert zwar noch, sollte aber nicht mehr genutzt werden. Stattdessen kann man ein eigenes Netzwerk anlegen, über das die Container dann abgeschirmt kommunizieren. Dies wird weiter unten im Kapitel „Docker Images für ein WebLogic Cluster“ beschrieben.

Bis zur Docker Version 1.12 gab es den sog. „Docker Swarm“[4] um eine Multi-Host Docker Umgebung einzurichten. Dessen durchaus komplexe Konfiguration wird durch das Tool „docker-machine“ stark vereinfacht. „docker-machine“ konfiguriert eine Reihe von virtuellen Maschinen (auf Basis von VirtualBox, AWS, Goolge CE, etc) oder nutzt bereits vorhandene Rechner, um einen Swarm – also ein Cluster von Docker Containern – anzulegen. Hierbei werden alle notwendigen Komponenten, wie z.B. auch das overlay Netzwerk, automatisch berücksichtigt.

War „Docker Swarm“ noch ein eigenständiges Tool, gibt es seit der Docker Version 1.12 den „Docker Swarm Mode“[5], der das Einrichten und Verwalten eines Docker Clusters weiter vereinfacht. Insbesondere benötigt man kein „externes“ Werkzeug wie docker-machine mehr. Stattdessen wird einfach mindestens ein Host zum Swarm Master erklärt und eine Reihe weiterer Rechner zu Swarm Worker Nodes. Der Docker Swarm Mode unterstützt nicht nur das Einrichten von Docker Clustern, sondern darüber hinaus auch die Orchestrierung komplexer Docker Applikationen (z.B. Web-Tier <-> Applikation <-> Datenbank) über Docker Services und Docker Stack.

Oracle und Docker

Docker ist auch Bestandteil der Linux Distribution von Oracle und im Rahmen des Oracle Linux Supports unterstützt. Darüber hinaus sind viele Oracle Produkte wie WebLogic, RDBMS, Tuxedo, etc. für Docker – z.T. auch auf anderen Linux Distributionen – zertifiziert. Vorgefertigte Docker Images dieser Produkte können – nach Anmeldung und Bestätigung der Lizenzvereinbarung – aus dem Docker Store[6] heruntergeladen werden. Zurzeit findet man dort u.a. ein Image mit der WebLogic Version 12.2.1.2. Dieses beinhaltet eine quasi leere WebLogic Domäne, die nur aus dem AdminServer besteht. Um eine „brauchbare" Domäne mit mehreren Managed Servern, einer deployten Applikation und den von ihr benötigten Ressourcen (wie DataSources, JMS Objekte, etc.) zu bekommen, muss dieses Image erweitert werden.

Beispiele hierfür findet man bei den Oracle Docker Images auf github[7] in dem Unterverzeichnis „OracleWebLogic". Hier liegen Anleitungen für das Bauen komplexer WebLogic Images – mit unterschiedlichen WebLogic Versionen und für unterschiedliche Topologien.

Bauen eines Docker Images mit einer WebLogic Domäne

Es gibt mehrere Möglichkeiten, eine WebLogic Domäne in einem Docker Image zu erstellen. Die einfachste Variante ist der bereits erwähnte Download des fertigen WebLogic Images vom Docker Store:

```
$ docker login

$ docker pull store/oracle/weblogic:12.2.1.2

$ docker images
REPOSITORY             TAG       IMAGE ID      CREATED      SIZE
store/oracle/weblogic  12.2.1.2  6dd42260529d  8 weeks ago  1.12 GB
```

Damit hat man nun in seiner lokalen Docker Umgebung ein WebLogic Image, das im Folgenden erweitert werden kann. Der Nachteil dieser Vorgehensweise ist jedoch die Beschränkung auf die in dem Image gepackte Version des WebLogic Servers, des Betriebssystems und des JDKs.

Diverse Beispiele für das Erzeugen und Erweitern einer WebLogic Domäne finden sich auf github.com[8]. Im Folgenden wird ein kurzes Beispiel gegeben, wie das hier heruntergeladene WebLogic Image, das ja noch keine Domänenkonfiguration beinhaltet, erweitert werden kann.

Erst mit dem ersten starten eines Containers von diesem Image wird eine Domäne erzeugt. Dies lässt sich an dem Image durch folgenden Docker Befehl überprüfen:

```
$ docker inspect store/oracle/weblogic:12.2.1.2 | grep CMD
    "CMD [\"/u01/oracle/createAndStartEmptyDomain.sh\"]"
```

Beim Starten des Containers wird also standardmäßig der Befehl „createAndStartEmptyDomain.sh" ausgeführt. Dieses Skript überprüft, ob der Container zum ersten Mal gestartet wird. Ist dies der Fall, wird zunächst eine neue Domäne angelegt und dann der Admin Server gestartet. Andernfalls ist die Domäne ja bereits vorhanden und es wird nur der Admin Server hochgefahren. Den Inhalt dieser Datei kann man sich auf der oben erwähnten github.com Seite ansehen. Oder aber man startet den Container mit einem alternativen Befehl, so dass nicht die Domäne angelegt, sondern nur der Inhalt des Skripts ausgegeben wird:

```
$ docker run --rm -it store/oracle/weblogic:12.2.1.2 \
  cat createAndStartEmptyDomain.sh
```

Es ist also wichtig zu verstehen: das aus dem Oracle Docker Store heruntergeladene Image enthält keine Domäne. Eine Domäne wird erst mit dem Starten des Containers angelegt. Sie befindet sich also nur in dem Container (und nicht im Image). Wird der Container gelöscht, ist auch die Domäne nicht mehr vorhanden. Da nur Images und nicht Container zwischen unterschiedlichen Umgebungen transportiert werden können, hat mein keine Möglichkeit, eine Domäne, die nicht in einem Image angelegt ist, auf andere Rechner zu transferieren.

Möchte man auf Basis dieses Images ein neues Image anlegen, in dem eine WebLogic Domäne, ggf. weitere Managed Server und vielleicht eine Applikation deployt sind, kann man die WLST Skripte in den oben erwähnten github Beispielen[8] als Vorlage benutzen. Abb. 2 zeigt Beispielhaft, wie dies funktioniert:

```
FROM store/oracle/weblogic:12.2.1.2
ARG ADMIN_PASSWORD
ENV ADMIN_USERNAME=weblogic
USER root
COPY container_scripts/* /u01/oracle/
RUN chown oracle:oracle *
USER oracle
RUN ./createWLSDomain.sh
EXPOSE 5556 7001 8001
CMD ["/u01/oracle/user_projects/domains/base_domain/startWebLogic.sh"]
```

Abb. 2: Beispiel eines Dockerfiles für den Aufbau einer WebLogic Domäne

Das hier dargestellte „Dockerfile“ ist eine Datei, vergleichbar mit einem c++ Makefile oder Maven build.xml, in der sequenziell die Befehle stehen, mit denen das neue Image angelegt werden soll. Hier wird also basierend auf dem aus dem Docker Store heruntergeladenen Image (FROM) ein neues Image gebaut. Das Passwort des Admin Servers wird dabei über einen Parameter beim Aufruf mitgegeben (ARG), der Admin-Username dagegen über eine Environment-Variable fest definiert (ENV). Zunächst werden dann die benötigten Skripte aus einem lokalen Verzeichnis in das Docker Image hineinkopiert (COPY), deren Berechtigungen angepasst (RUN) und mit diesen Skripten unter dem User oracle (USER) eine neue Domäne angelegt (RUN). Anschließend werden die NodeManager, AdminServer und Managed Server Ports bekannt gemacht (EXPOSE) und der Default-Befehl festgelegt, der beim Starten des Containers ausgeführt werden soll (CMD).

Dieses Image wird durch folgenden Befehl gebaut:

```
$ docker build --build-arg ADMIN_PASSWORD=welcome1 -t wls-domain:12.2.1.2 .
```

Der Punkt am Ende der Befehlszeile bedeutet, dass das Dockerfile aus dem aktuellen Verzeichnis benutzt werden soll. Mit dem Befehl

```
$ docker run -d -p 4711:7001 --name wlsadmin wls-domain:12.2.1.2
```

wird nun der AdminServer im Docker Container gestartet. Dieser ist dann über den Host-Rechner und dessen Port 4711 erreichbar.

Mit dem selben Image können nun weitere Docker Container mit Managed Servern gestartet werden. Dafür muss nur der default Befehl (CMD), der ja den Admin Server startet, durch den Aufruf eines

anderen Shell-Skripts zum Anlegen und Starten eines Managed Servers ersetzt werden. Dieses Shell-Skript muss sich natürlich ebenfalls bereits im Image befinden, muss also im Rahmen des COPY Befehls beim Anlegen des Images mit kopiert worden sein.

Allerdings stößt man nun auf ein Problem. Denn der Managed Server, der in diesem neuen Container gestartet werden soll, versucht ja auf den WebLogic AdminServer zuzugreifen. Wie bereits erwähnt sind Docker Container aber derart isoliert, dass ein Netzwerk-Zugriff auf einen anderen Docker Container normalerweise nicht möglich ist.

Docker Images für ein WebLogic Cluster

Damit das oben erwähnte Beispiel funktioniert, muss also ein Netzwerk für den Zugriff zwischen Docker Containern konfiguriert werden. Laufen alle Container auf einem Host, genügt hierfür ein „bridge“ Netzwerk, ggf. mit einem dedizierten Subnetz:

```
docker network create --subnet 192.168.168.0/24 my-wls-network
```

Werden jetzt alle Container mit diesem neuen Netzwerk (und ggf. einer festen IP Adresse) gestartet, ist eine Verbindung mit dem Admin Server problemlos möglich:

```
docker run -d -e ADMIN_PASSWORD=welcome1 -p 5001:8001 -h ms1 --name ms1 \
           --network my-wls-network --ip 192.168.168.3 \
           wls-domain:12.2.1.2 ./createServer.sh
```

Das “bridge” Netzwerk funktioniert allerdings nur für Container auf einem Host. Möchte man ein Cluster von Docker Containern über mehrere Rechner hinweg betreiben, muss stattdessen ein sogenanntes „overlay“-Netzwerk eingerichtet werden. Dies wird durch den Docker Swarm Mode deutlich vereinfacht. Hierfür kürt man (mindestens) einen Host zum Swarm Master. Die dabei publizierte IP Adresse (--advertise-addr) muss für alle anderen Rechner, auf denen Docker Worker Nodes laufen sollen, erreichbar sein:

```
$ docker swarm init --advertise-addr <ip-address>
```

Dieses Kommando gibt ein Token zurück, mit dem nun beliebig viele andere Rechner zu Swarm Worker Nodes gemacht werden können:

```
$ docker swarm join --token <token aus dem init Befehl> <ip-address>:2377
```

Im swarm mode werden nun nicht mehr einzelne Container gestartet. Stattdessen werden Docker Services angelegt und der Swarm Mode sorgt für die Verteilung und Überwachung der an dem Service beteiligten Container. Dafür muss das Image, das von den Services benutzt wird, von allen Docker Swarm Nodes erreichbar sein. Deshalb muss es entweder auf docker hub abgelegt werden oder man startet eine lokale Registry, auf die alle Swarm Nodes zugreifen können. In diesem Beispiel wird eine lokale Registry auf dem Host „docker-registry“ gestartet (mit dem default-Port 5000). Nach dem „push“ des Images in diese Registry kann der Docker Service erstellt werden:

```
$ docker service create --name wlsadmin --network my-wls-overlay-network \
  --publish 4567:7001 my-docker-registry:5000/wls-domain:12.2.1.2
```

Sobald der WebLogic Admin Server läuft, wird der Service für die Managed Server gestartet – zunächst mit nur einem Container:

```
$ docker service create --name wlsms --network my-wls-overlay-network \
  --publish 5678:8001 my-docker-registry:5000/wls-domain:12.2.1.2 \
  ./createServer.sh
```

Der Befehl

```
$ docker service scale wlsms=3
```

startet weitere Managed Server Container. Der Swarm Mode sorgt hier nicht nur für das Verteilen der Services auf die beteiligten Knoten, sondern auch für das automatische Nachstarten ausgefallener Container.

Oracle Container Cloud Services

Natürlich kann man die so erstellten Docker Container auch in der Oracle Cloud betreiben. Neben der nativen Installation von Docker in Oracle Compute Cloud Services bietet sich die Nutzung der Oracle Container Cloud Services an. Hierbei handelt es sich um eine Administrations-Oberfläche, die Oracle als Service anbietet, um komplexe Docker Container Infrastrukturen und Docker Stacks – auch als Cluster – zu betreiben. Das Deployment von „Stacks“ in Container Cloud Services ist weitgehend kompatibel zum Deployment von Docker Compose. Bei Docker Compose werden die Abhängigkeiten und die Ressourcen der verschiedenen Komponenten in sog. YAML Files beschrieben. Diese YAML Files können dann in den Container Cloud Service kopiert werden, um diese Konfiguration als „Stack“ dort deployen zu lassen.

```
version: '3'
networks:
  wlsnet:
    driver: bridge
services:
  wlsadmin:
    image: wlsimage
    build:
      context: .
      args:
        ADMIN_PASSWORD: welcome1
    ports:
      - "4567:7001"
    networks:
      - wlsnet
    hostname: wlsadmin
  ms1:
    image: wlsimage
    build:
      context: .
    depends_on:
      - wlsadmin
    ports:
      - "5001:8001"
    command: /u01/oracle/createServer.sh
    environment:
      - ADMIN_PASSWORD=welcome1
    networks:
      - wlsnet
    hostname: ms1
```

Abb. 3: Beispiel eines YAML Files für das Deployment eines WLS Domänen-Stacks

Dieses Beispiel zeigt eine vergleichbare Service Architektur, wie die zuvor beschriebene. Hier werden alle Container gemeinsam deployt. Abhängigkeiten sind übersichtlich dargestellt. So lassen sich komplexe Container-Architekturen entweder als Docker Compose auf einem Host, als Docker Stack im Swarm Mode in einem Rechner-Cluster oder als Cloud Service in der Oracle Container Cloud deployen.

Kontaktadresse:

Thomas Robert

Oracle Deutschland B.V. & Co. KG

Kühnehöfe 5

22769 Hamburg

Telefon: +49 (0) 40-89091188

Fax: +49 (0) 40-89091250

E-Mail thomas.robert@oracle.com

Internet: www.oracle.com

---

[1] https://www.oracle.com/de/corporate/pressrelease/oracle-databases-tools-docker-store-20170420.html

[2] https://www.oracle.com/de/middleware/weblogic

[3] https://de.wikipedia.org/wiki/Docker_(Software)

[4] https://docs.docker.com/swarm/overview/

[5] https://docs.docker.com/engine/swarm/

[6] https://store.docker.com

[7] https://github.com/oracle/docker-images

[8] https://github.com/oracle/docker-images/tree/master/OracleWebLogic/samples