Docker und Oracle's JDK – Grundlagen für Entwickler

Torsten Grünen, Peter Kerekes, Andreas Chatziantoniou
Devtops Gmbh
Büchenbach

Schlüsselworte
Docker, JDK, Entwicklung

Einleitung

Wer in Projekten die Notwendigkeit hat, um schnell und zuverlässig Umgebungen bereit zu stellen und diese auch mit übersichtlichen Anwendungen zu "bestücken" wird immer öfter auf Docker stoßen. Was aber ist Docker nun genau? Warum hilft es mir, um meinen Code zum Laufen zu bringen. Kann ich unbegrenzt skalieren? Wie sieht das im Betrieb aus? Dieser Vortrag zeigt in kleinen Beispielen wie ein Docker Image entsteht, wie es mit Leben gefüllt wird und woran ich als Entwickler denken muss wenn mein Docker Container verschwunden ist.

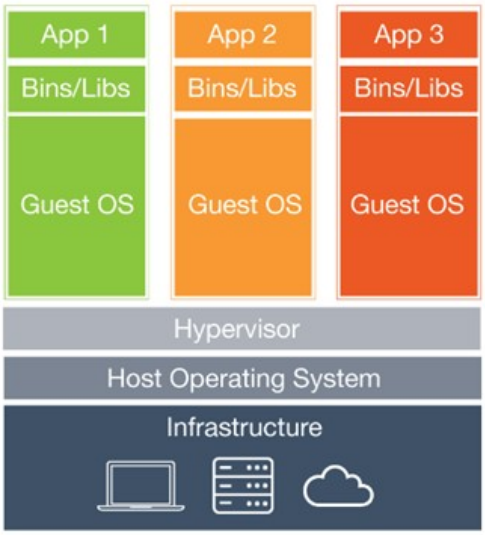
Der Vortrag hat einen Fokus auf Java-Anwendungen, zeigt aber auch wie bestehende Fusion-Middleware-Umgebungen von Docker profitieren können stellt aber gleichzeitig die Frage nach den Grenzen von Docker im klassischen Application-Server-Umfeld.

Was ist Docker?

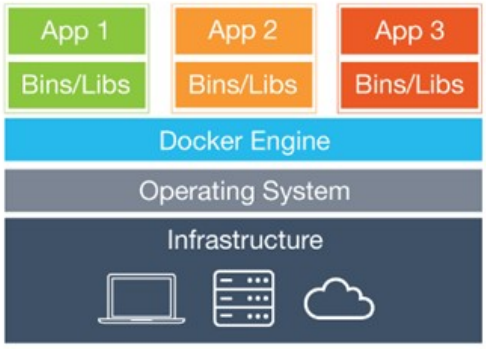
Von der Docker Seite:

Was ist ein Container?

Container sind eine Art und Weise um Software so bereitzustellen (paketieren), dass diese isoliert auf einem geteilten Operating System laufen zu lassen.



Virtual Machines



Containers

Im Gegensatz zu Virtuellen Maschinen haben Container kein eigenes Operating System. Ein Container enthält nur Libraries und Konfigurationen die notwendig sind um die Software zum Laufen zu bringen.

Hierdurch erhält man mit Docker ein effizientes, kleines und abgeschottetes System, dass immer lauffähig ist, unabhängig davon, wo es eingespielt wurde.
Um genau zu sein, die Docker Container laufen simultan auf einem Linux Rechner und teilen sich die vorhandenen Ressourcen. Ohne Docker all diese Ressources hätten mehrmals zur Verfügung gestellt werden müssen.

Warum hilft es mir, um meinen Code zum Laufen zu bringen?

Um Software zu Laufen zu bringen, muss die Umgebung so konfiguriert sein, dass die Software korrekt eingesetzt werden kann. Dies ist eine Aufgabe, die, je nach Komplexität der Umgebung, zwischen ein paar Schritten bis hin zu einigen Hundert Schritten erfordern.
Dies wird am besten sichtbar, wenn wir eine Oracle Fusion Middleware Umgebung betrachten. Zuerst brauchen wir (auf einem vorhandenen Operating System) eine Installation des WebLogic Servers. Dieser kann - wie wir wissen - auf verschiedene Arten installiert werden.

Nun kommt es zum Aufbau einer Domäne. Hier gibt es so vielfältige Konfigurationsmöglichkeiten, dass es sinnlos ist diese zählen zu wollen.

Die Problematik entsteht nun an der folgenden Stelle: Es gibt Konfigurationen, welche die Lauffähigkeit meiner Software direkt beeinflussen, und solche, die nur geringfügige Auswirkungen haben.

Als Beispiel können wir hier die Konfiguration der JVM Heapsize und die Konfiguration der Logfiles betrachten. Wenn meine JVM nicht korrekt konfiguriert ist (im Allgemeinen: zu klein), dann wird meine JEE Anwendung im WebLogic Server nicht laufen. Wenn ich mein Logfile statt zeitbasiert nun größenbasiert rotiere wird meine Anwendung keine Probleme damit haben.

Standardisierung

Dies ist ein starkes Plädoyer für die Automatisierung und Festschreibung einer Konfiguration. Und genau hier setzt Docker auf. Interessanter wird dies genau dann, wenn ich alle Umgebungen meines Produktes betrachte. Idealerweise sind alle Konfigurationen in allen Umgebungen identisch. Die "Ausrede" des Entwicklers "es funktioniert aber auf meiner eigenen Maschine" kommt dann nicht mehr zum Tragen.

Die Erfahrung zeigt, dass ein Entwickler an "seiner eigenen Maschine" am besten arbeitet. Dies kommt dadurch, dass er dort seine eigenen "Customizations" hat. Genau seine Einstellung die er braucht, die für ihn am besten funktioniert. Dies hat sehr oft zu Situationen geführt in der Fehler entstanden sind, weil eine bestimmte Library mit einer speziellen Version nicht oder an einer "merkwürdigen" Stelle im Classpath stand.

Wenn wir diesen Schmerz nun nehmen und mit einem standardisierten Image arbeiten entfällt dieser Aspekt. Da wir zu einem späteren Zeitpunkt dieses Image mehrmals benutzen (100/1000/1000000) ist es absolut notwendig um sich an die Vorgaben zu halten und in genau diesem festgelegten Umfeld zu entwickeln. Docker hilft daher nicht nur beim Entwickeln, sondern auch beim Produktionsbetrieb. Denn die Produktionsumgebung ist IDENTISCH mit der Entwicklungsumgebung!

Komplexe Systeme entstehen auch, da für verschiedene Aspekte verschiedene Tools, Sprachen und gesonderte Konfigurationen benutzt werden. Wer jemals auf seinem Rechner versucht hat mit zwei verschiedenen Versionen von Java zu arbeiten oder wer das Vergnügen hatte um eine Produktionsumgebung mit zwei Java Versionen parallel betreiben zu dürfen kann ein langes Klagelied singen. Docker sorgt hier für Abhilfe: Für Version XY123 nehme ich DockerImageXY123 und für Version GH987 nehme ich DockerImageGH987. Fertig.

Einer der interessanten Aspekte für Entwickler und Betreiber ist die Tatsache, dass ich ein Docker Image sehr schnell zu einer früheren Version zurück rollen kann. Hiermit steht im Fehlerfall eine Möglichkeit zur Verfügung schnell zum Ursprünglichen Zustand zurück zu kehren.

Best Practices

Für Entwickler gibt es nun zwei Best Practices die das Entwickeln einfacher machen:

- Stelle sicher, dass alle Abhängigkeiten im gleichen Arbeitsverzeichnis stehen wie der Code.
- Sorge dafür, dass alle Compile und Run-Kommandos innerhalb des Docker Containers laufen.

Es gibt aber auch ein paar Anti-Patterns bei denen man als Entwickler vorsichtig sein muss.

Container sind unveränderlich. Wenn der Container angepasst werden muss dann müssen alle vorhandenen Container erneuert werden.
Speichere keine Daten im Container. Wenn dieser gestoppt, zerstört oder ersetzt wird sind die Daten weg
Benutze kein Ein-Lagen-Image. Baue Images für das OS, andere für die Benutzernamendefinition, eines für die Runtime usw.
Mache keine Kopien von Docker Images - baue sie neu
Lass Prozesse nicht als root laufen - eine Sicherheitslücke
Finger weg von IP Adressen - benutze Environmentvariablen um Hostnamen und Ports zu benutzen
Sorge für ein vernünftiges Logging - nicht im Image

Wenn es dann in Produktion geht ist schon etwas Vorbereitung notwendig. Insbesondere bei der Netzwerkkonfiguration ist es sinnvoll um vor dem Start nachzudenken wie man sein Netzwerk - das ja tausende von Container besitzen kann - einrichten will und muss.

Kontaktadresse:
Torsten Grünen
DevTops GmbH
Ringstr. 13
D- 91186 Büchenbach

Telefon: +49 (0) 12-345 6789
Fax: +49 (0) 12-345 6788
E-Mail torsten.gruenen@devtops.gmbh
Internet: www.devtops.gmbh