Docker, Möglichkeiten und Fallstricke der Containerisierung von Legacy Software

Andreas Scheuer
htw saar – Hochschule für Technik und Wirtschaft des Saarlandes
Seminar “Angewandte Informatik”
Sommersemester 2022

***Zusammenfassung*—In dieser Arbeit werden die Problematiken bei der Containerisierung von Windows Legacy - Software erläutert. Hierbei wird aufgezeigt welche Anforderungen an die zu containerisierende Software bestehen, sowie die Restriktionen und Problematiken innerhalb eines Windows Umfeld auftreten.**

I. Einleitung

Aufgrund gewünschten Portabilität oder Skalierbarkeit kann es möglich sein das der Wunsch besteht Legacy Software zu containerisieren. In diesem Kapitel wird kurz auf die begriffe Legacy sowie Containerisierung eingegangen.

1) Legacy - Software: Der Begriff Legacy - Software wird in der Informatik oft im engeren Sinne mit einer historisch gewachsenen Anwendung und oder als Altlast, Hinterlassenschaft verwendet. (Michael C Feathers) schrieb dazu: Das in der Branche Legacy Code oftmals als schwer zu änderbaren oder nicht verständlicher Code bezeichnet wird. [1] Dies kann dann z.B auftreten wenn man aus kostengründe software einkauft welche nicht quelloffen ist welche dann mit den jahren nicht an die eigenen anforderungen angepasst wird oder aus oben genannten kostengründen nicht angepasst werden soll.

2) Containerisierung: die containerisierung wird in der informatik verwendet um die virtualisierung einer laufzeitumgebung mittels software zu gewährleisten. hierfür wird oftmals Docker verwendet, einem containerisierungs tool welches container erstellen kann die alle notwendige abhängigkeiten besitzen um software unabhängig zu betreiben. die funktionsweise von docker lässt sich folgendermaßen erklären, anstelle wie bei virtuellen maschinen die einen hypervisor benutzen um die befehle des instanzierten betriebssystems übersetzen auf die des host systems, laufen container direkt auf dem host system.

Ein weitere Vorteil von Container ist das man sie so konfigurieren kann das sie ausschließlich für die Software benötigten Abhängigkeiten besitzt und nicht den ganzen overhead einer virtualisierung. Die erstellung solcher container erfolgt mittels images, dieses beinhalte read-only informationen die für die erstellung eines containers notwendig sind. images werden in layer definiert. Diese layer kann man in einer dockerfile definieren, jeder ausgeführte befehl wird als zusätzlicher layer dem image hinzugefügt und es wird ein hashwert gezogen für den späteren gebrauch.

Wird nun also so ein Image erstellt schaut erstmal docker in seinem build cache ob die layer die benötigt werden bereits

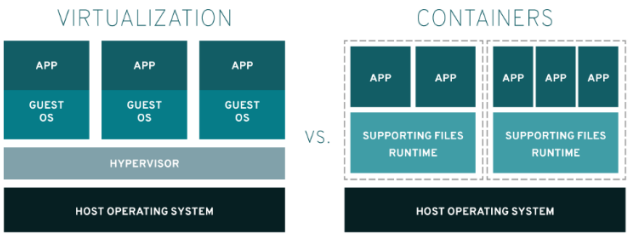

Abbildung 1. Unterschied zwischen Virtualisierung und Container, Quelle: Container vs VM [2]

```
Dockerfile
1  FROM openjdk:20-slim-buster
2  EXPOSE 8080
3  COPY target/demo-0.0.1-SNAPSHOT.jar app.jar
4  ENTRYPOINT ["java","-jar","/app.jar"]
```

Abbildung 2. Beispiel einer Dockerfile

als hash vorhanden sind und können so wieder verwendet werden. Weitere informationen lassen sich dies bezüglich aus der Overview of Docker Build [3] nachschlagen.

II. Szenario

oftmals kann es dazu kommen das man Legacy - Software im betrieb hat welche mehrere Jahre alt ist und teils nicht mehr gewartet wird oder nicht mit den wachsenden anforderungen wächst. ist diese dann auch noch eine systemkritische komponente kann sie schnell zu einem bottleneck in einem workflow werden. das ersetzen solcher software lässt sich je nach umfang nur mit einem hohen geldaufwand bewerkstelligen. Da die software weder quelloffen ist noch einfach ausgetauscht werden kann liegt hier der versuch nahe sie zu containerisieren um sie dann anschließend zu skalieren. dies wäre natürlich auch mittels virtuellen maschinen möglich hierbei ist aber zu beachten das dies einen erhöhten konfigurationsaufwand hätte.

III. Auftretende Probleme

????

IV. Lösungsansätze

untersetion

V. Zusammenfassung und Ausblick

Zusammenfassung

VI. Zur Person

Name: Andreas Scheuer
Matrikelnummer: 3849139
Studiengang: Praktische Informatik
Mail: pib.andreas.scheuer@htwsaar.de

Literatur

[1] M. C. Feathers, *Effektives Arbeiten mit Legacy Code: Refactoring und Testen bestehender Software*. BoD–Books on Demand, 2020.
[2] Redhat, "Containers vs vm," Website, online erhältlich unter https://www.redhat.com/de/topics/containers/containers-vs-vms; abgerufen am 29. November 2022.
[3] I. Docker, "Overview of docker build," Website, online erhältlich unter https://docs.docker.com/build/; abgerufen am 29. November 2022.