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 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

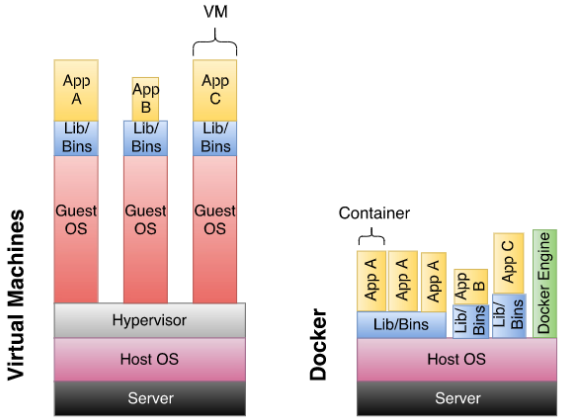

Abbildung 1. Unterschied zwischen Virtualisierung und Container, Quelle: Albin Sundqvist [2]

```
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

aufgrund der komplexiät solcher szenarien ist es mir jetzt nicht möglich auf alle problematiken einzugehen, dennoch möchte ich hier die häufigsten aufgreifen. während der containerisierung kann man auf diverse probleme stoßen die mitunter den ganzen vorgang erschweren oder sogar unmöglich werden lassen. um dies zu verhindern sollte man folgende punkte stets kritisch prüfen und schauen ob diese gegen eine containerisierung sprechen.

Source Code:

Bei Legacy Software kann der Source Code in den unterschiedlichsten varianten vorliegen, manchmal auch garnicht. Hier eine auflistung der möglichkeiten.

(a) Er ist vorliegend, man hat sogar die möglichkeiten änderungen vorzunehmen.

(b) teilweise oder ganz vorliegend aber keine mögichkeit änderungen vorzunehmen.

(c) Source Code liegt nicht vor und somit weder änderbar noch einsehbar.

Arbeitsweise der Software:

Auch hier gibt es einige punkte die gegen eine Containerisierung sprechen, gerade desktop applications die auf einem frontend aufbauen das zu bedienen gilt fallen hier heraus. der mehraufwand ein solches system in betrieb zu nehmen wäre höher als das aufsetzen einer ganzen VM. daher eignen sich eher Applications die entweder über command line argumente annehmen oder beim start ihre arbeit von selbst verrichten ohne weiteres zutun.

Installation:

Hier gilt ähnlich bei der Arbeitsweise der Software, eine Software lässt sich sofern sie eine Installation benötigt, nur mit einem erhöhtem Mehraufwand installieren wenn es dafür nur eine grafische oberfläche gibt die man bedienen muss. Hierfür müssten dann alle Installationsschritte händisch (oder mittels script) nachgestellt werden. Gerade bei einem Betriebssystem wie Windows kann dies zu einer äußerst mühsehligen arbeitn werden.

Spezielle Abhängigkeiten:

Je nach Software kann es sein das spezielle abhängigkeiten vorhanden sind wie z.B

lizensen:

Dies ist oft der fall wenn die Software eingekauft wurde. diese können oder müssen entsprechend in den Container integriert werden.

Hardware Abhängigkeiten:

Es kann auch schonmal vorkommen das gerade bei gekaufter Software lizensen anhand von diversen Hardware Konfigurationen, Mac Adressen, oder sonstigem erstellt werden und dann nur für diesen Computer freigegeben sind. Dies sind besonders schwierige fälle für die es aber auch Lösungsansätze gibt.

Wie und Wo werden Daten gespeichert:

Oftmals verarbeiten Programme ja nicht nur Daten sondern speichern diese dann auch, das muss entsprechend im vorfeld klar sein wie und wo die daten gespeichert werden. gerade programme die lokal ihre daten speichern und abrufen sind mit einer standartisierten containerisierung problematisch da container in sich ja zustandslos sind und bei jedem neustart immer wieder den urzustand wieder erhalten.

Uvm...

wie schon oben beschrieben, gibt es einige mehrere probleme die auftreten können und aufgrund der komplexität dessen werde ich nur ein paar Lösungsansätze für die oben beschrieben probleme angehen.

IV. Lösungsansätze

Source Code

Quelloffen:

Dies ist sehr unproblematisch, man kann das Verhalten der Software nachvollziehen und ggf sogar ändern.

Nicht Quelloffen aber Dokumentiert:

Hier lassen sich keine zwar keine änderungen vornehmen aber man kann entsprechend der Dokumentationsqualität das verhalten nachvollziehen und sich darauf entsprechend einstellen bzw. den Container entsprechend anpassen.

Nicht Quelloffen und nicht Dokumentiert:

Da weder Sourcecode nocht Dokumentation vorhanden ist, gibt es hier leider nur zwei möglichkeiten. Den hersteller zu rate ziehen oder durch trial and error oder gar durch reverse engineering [4] (achtung kann strafrechtliche folgen mit sich ziehen) das verhalten des programmes zu ergründen.

Arbeitsweise der Software

Wie schon bereits im unterpunkt **Auftretende**

Probleme beschrieben macht es natürlich wenig sind software zu containerisieren die trotzdem noch händisch bedient werden muss. dies ist auch ein klarer auscheidepunkt für die containerisierung da ja immernoch personal benötigt wird um es zu bedienen und der mehraufwand dies zu bewerkstelligen höher als der nutzen wäre. dennoch möchte ich hier kurz eine möglichkeiten aufzeigen. Software die ein Webfrontend besitzen kann man natürlich nicht nur prima containerisierung sondern auch mittels Puppeteer [5] vollautomatisch bedienen lassen.

Installation

Auch hier ist es äußerst problematisch wenn software über eine installations routine verfügt durch die man sich durch klicken muss. gerade im windows umfeld erlebt man dies ja sehr häufig. abhilfe dafür gibt es, gerade auch windows software besitzt oft eine möglichkeit der installation via command line interface. Für die Windows Microsoft Standard Installer in kurz msi files gibt es folgende Website die man zu rate ziehen kann:**Microsoft Standard Installer Command-Line Optionen** [6] oder folgende seite die nicht nur Microsoft bezogen ist: **Silent Install HQ** [7] die einem mit genügend Tutorials versorgt um die entsprechende Software dann doch zu installieren.

Spezielle Abhängigkeiten

Wie und Wo werden Daten gespeichert

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] A. Sundqvist, "Guidelines to convert legacy applications into docker container services," 2020.
[3] I. Docker, "Overview of docker build," Website, online erhältlich unter https://docs.docker.com/build/; abgerufen am 16. Januar 2023.
[4] E. Eilam, *Reversing: secrets of reverse engineering*. John Wiley & Sons, 2011.
[5] I. Google, "Puppeteer," Website, online erhältlich unter https://pptr.dev; abgerufen am 16. Januar 2023.
[6] Microsoft, "Microsoft standard installer command-line optionen," Website, online erhältlich unter https://learn.microsoft.com/de-de/windows/win32/msi/standard-installer-command-line-options; abgerufen am 16. Januar 2023.
[7] J. Bergner, "Silent install hq," Website, online erhältlich unter https://silentinstallhq.com; abgerufen am 16. Januar 2023.