Fragen der Teilnehmer im Webcast Heise/InterSystems „Gewinnbringend und risikofrei: Containerisierung ohne Reue“ am 19.5.2020

Hier sind die Antworten zu den in der Sendung unbeantwortet gebliebenen Fragen:

1) Als Grundlage sollte ein beschriebener E2E-Business-Prozess dienen. Dann extrapolieren und dann kann man sich ja über Microservices Gedanken machen. Ist an dieser Sichtweise etwas falsch?

Antwort: Das ist sicherlich eine valide Vorgehensweise.

2) Ich bin schon lange aus Application Development draußen. Aber was hat sich eigentlich am SW-Entwicklungsprozess geändert? Von Requirements > Design (APPL, API; DB) > Development > Function Testing > Integration Testing > Performance Testing > Acceptance Test > EU-Educaton > Deploy & Operations. Alles auf Basis eines Datenflussmodells und Datenbankdesigns.

Antwort: Der Prozess ist alles in allem sicher weiterhin gültig. Die Art und Weise, wie die einzelnen Prozessschritte umgesetzt werden und wie die Orchestrierung der einzelnen Schritte erfolgt, unterliegt sicher einer gewissen Dynamik und Veränderung.

3) Ist der Lastgenerator im Standard Docker enthalten?

Antwort: Nein, der Lastgenerator, den Sie in der Demo gesehen haben, war in InterSystems IRIS implementiert. Es gibt aber verfügbare Lastgeneratoren, um die API zu „stressen“. SOAPUI bietet beispielsweise auch etwas. Weitere Optionen:

RESTful Stress: https://chrome.google.com/webstore/detail/restful-stress/lljgneahfmgjmpglpbhmkangancgdgeb

awesome-http-benchmark: https://github.com/denji/awesome-http-benchmark

4) Inwieweit hat DevOps was mit Servicevirtualiserung und Parallelentwicklung zu tun? Hype ist ja heute agile und nicht Waterfall.

Antwort: Service Virtualisierung und Parallelentwicklung sind Bausteine, die bei der Umsetzung des DevOps Ansatzes eingesetzt bzw. genutzt werden können.

5) Woher weiß das System, welches Image sich hinter msorder2 verbirgt? Hat ja einen anderen Namen, als der ursprüngliche Container.

Antwort: Ich hatte den Container vorab bereits erzeugt, ihn aber gestoppt. Ein Container wird mit dem docker run Kommando erzeugt. Einer der Parameter bei diesem Kommando ist der Name des Images, zu dem der Container erzeugt werden soll.

6) Kann eine DLL oder SO als Container gesehen?

Antwort: Ein Container ist zur Laufzeit im Prinzip ein Prozess auf dem Host. Dieser Prozess stellt entweder die Applikation selber da oder startet weitere Prozesse auf dem Host, die dann die Applikation ausmachen. Wird der Vaterprozess beendet, wird auch der Container beendet.

Auf der Disk ist der Container im Prinzip ein Stück Filesystem, in dem der benötigte Code z.B. in einem Binary liegt und eben alle benötigten DLLs oder SOs.

7) Ist application-source ein Teil von Image? Wenn ich docker als dev-enviroment benutze, ist es mir bequemer, natürlich source zu mounten (um image nicht neu builden).

Antwort: Theoretisch können Sie auch den Sourcecode in den Container legen, aber in der Praxis sieht der Entwicklungsprozess eher so aus, dass die Entwickler z.B. ein lokales Git Repository haben und dann ihre Änderungen ins Git committen. D.h. die Quelle der Wahrheit stellt das Source-Control System da.

Container sind eine Deployment-Option und meiner Meinung nach nicht so sehr eine Entwicklungsumgebung.

8) Wie schaut es mit den Berechtigungen aus? D.h. unter welchen credentials laufen die container? Alle unter denselben creds oder lässt sich das steuern?

Antwort: Dies lässt sich beeinflussen. Die InterSystems IRIS Container laufen z.B. nicht als Root User und bei default sind Container auch nicht privilegiert. Hier kann man aber Einfluss nehmen. Dies mag von Containerplattform zu Containerplattform variieren.

9) Läuft die API auch in einem Container?

Antwort: In meinem Falle ja.

10) Wie wirkt sich die Verwendung von Containern auf Lizenzfragen aus?

Antwort: Lizenzfragen für das Host-Betriebssystem? Der Container enthält ja kein Betriebssystem, sondern läuft wie eine Applikation auf dem Host. Der sollte natürlich ordentlich lizensiert sein. Der Container verursacht keine zusätzlichen Lizenzkosten im Vergleich zum Deployment ohne Container, es sei denn die Container-Runtime ist kostenpflichtig.

11) Welches API Management benutzt Ihr?

Antwort: InterSystems IRIS beinhaltet den InterSystems API-Manager. Den hatte ich auch in der Demo benutzt.

12) Bitte erklären Sie noch einmal den Zusammenhang zwischen den Container und Microservices - 1:1 oder 1:n oder n:1 ...?

Antwort: Die Verteilung liegt bei Ihnen. Ich hatte meine Service funktionsorientiert gebündelt und für jede dieser Einheiten einen eigenen Container.

Theoretisch können Sie aber auch jeden Endpunkt in einen eigenen Container legen. Damit bekommen Sie aber ein riesiges Cluster aus Containern, der kaum noch beherrschbar sein dürfte.

13) Müssen die Entwickler meiner Applikation in irgendeiner Weise auf zentrale Komponente API-Manager Rücksicht nehmen oder ist dieser völlig transparent?

Antwort: Der API-Manager (zumindest der InterSystems API-Manager) ist völlig transparent.

14) Was genau macht IRIS, bzw. was verwenden Leute, die es (noch) nicht haben? Und was macht IRIS dann leichter und besser?

Antwort: InterSystems IRIS ist eine Datenplattform, die hochgradig interoperabel ist und die es ermöglicht, regelbasierte Business-Prozesse mit Human-Workflow-Elementen systemübergreifend implementieren zu können. Darüber hinaus bietet es Analytics von strukturierten und unstrukturierten Daten, ergänzt um Machine-Learning-Verfahren. Ein weiterer Fokus liegt auf Skalierbarkeit. Sie können auf einem Notebook beginnen, wie ich ihn auch in meiner Demo hatte und wenn die Anforderungen wachsen, wächst Ihre Applikation mit und kann sowohl vertikal als auch horizontal skalieren.

15) Ist anstelle der Verwendung von REST Services auch die Verwendung von SOAP Services möglich?

Antwort: Ja. SOAP ist aber schwergewichtiger, allein durch die XML-Strukturen. Dafür ist es ein Standard. REST ist kein Standard und beruht letztlich auf Konvention, aber durch die Verwendung von JSON in der Regel schlanker. JSON ist aber kein MUSS bei REST.

16) Für welche Anwendungen ist Containerisierung absolut sinnvoll? Anzahl User-abhängig?

Antwort: Ich würde Containerisierung nicht zwingend von der User-Anzahl abhängig machen, sondern davon, ob ich ein häufiges Deployment meiner Applikation anstrebe oder ob ich einzelne Bestandteile der Applikation skalieren können will usw.