hochschule mannheim

Fakultät für Informatik
Hochschule Mannheim

Studienarbeit

Virtualisierte Arbeitsumgebung für den Test verteilter Systeme

Microservice-Architektur mit Docker, Kafka, Clojure & Elm

Jan-Philipp Willem

im Sommersemster 2017

Zusammenfassung

Ein verteiltes Softwaresystem zu entwickeln und ebenso dessen Auswirkungen zu verstehen, ist kein leichtes für einen durchschnittlichen Informatik-Studenten. Oft ist es der Fall, dass es schon anspruchsvoll genug ist, alleine die Grundlagen umzusetzen, welche es bei einer Aufgabe anzuwenden vermag. Um diese Hürde etwas zu erleichtern, soll eine Umgebung geschaffen werden, welche es ermöglicht, eine verteilte Aufgabe unabhängig des eigenen Computers zu testen. Eine Programmieraufgabe soll von einem Dozenten als ein Experiment definiert werden können, welches eine gegebene Anzahl an Instanzen besitzt und durch einen Studenten genutzt werden kann, die Ausgaben der erstellten Programme auf voneinander getrennten Rechnern zu beobachten.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Problemstellung

1.1 Bisheriger Ablauf

1.2 Anforderungen

Kapitel 2

Grundlagen

2.1 Funktionale Programmierung

2.2 Verwendete Programmiersprachen

2.2.1 Clojure

2.2.2 Elm

2.3 Eingesetzte Webtechnologien

2.3.1 Websockets

2.3.2 Single-Page-Applications

2.4 Microservices

2.5 Container-Virtualisierung mit Docker

Docker[1] ist eine Software zum Deployment von Applikationen innerhalb von Containern. Im Vergleich zu Virtuelle Maschinen (VM's), ähnelt sich die Vorgehensweise, jedoch laufen die Prozesse der Container direkt auf dem Host-Betriebssystem. Trotz dieser Tatsache sind die Prozesse mithilfe von unter Anderem (u.A.) Control-Groups und Kernel Namespaces voneinander isoliert. Weiterhin hat man die Möglichkeit mit gängigen Mandatory-Access-Control-Frameworks wie SE-Linux oder App-Armor die Rechte innerhalb der Container zu beschränken. Als Anforderung besteht keine spezifische Hardware-Infrastruktur wie bei beispielsweise VMWare ESXi. Ebenso ist Docker mittlerweile auf allen gängigen Betriebssystemen lauffähig, wobei Linux-Derivate die beste Unterstützung erhalten.

Ein Container wird mithilfe eines Dockerfiles beschrieben. Darin werden deskriptive Instruktionen definiert um Abhängigkeiten zu installieren, Konfigurationen vorzunehmen und andere Build-Steps auszuführen. Ein spezifischer Container wird als Image bezeichnet und setzt sich aus granularen Sub-Images

[1] Link zu Docker

zusammen. Beim Erzeugen von Images eigener Dockerfiles müssen im Vergleich zu VM's keine großen Dateien transferiert werden, da sie aus ihrem Rezept reproduzierbar sind. Es besteht auch die Möglichkeit, von anderen Dockerfiles zu erben und diese über das Netzwerk verteilt bereitzustellen. Falls die über eine Docker-Registry angebotenen Schichten eines Containers noch nicht auf dem eigenen Rechner vorhanden sind, so werden diese heruntergeladen.

Weiterhin gibt es einen entscheidenden Vorteil gegenüber einer traditionellen VM: Ein Dockerfile trägt implizit zur Dokumentation bei, da jede Änderung an einem Container in seinem Rezept ergänzt werden muss.

Wird eine Applikation in einzelnen Services aufgeteilt, so bietet sich das Tool **Docker-Compose**[2] an. Es ermöglicht in einer einzelnen Datei (`docker-compose.yml`) alle Konfigurations-Parameter der einzelnen Containern zu definieren. Das können beispielsweise (bspw.) Volume-Mounts, Ports, Environment-Variables, Netzwerke sein, welche sonst bei jedem `docker exec` Command angegeben werden müssten. Es sind so ebenso getrennte Environments für Development, Test und Production als zentrale Dokumentation denkbar.

2.6 Apache Kafka

[2]Link zu DC

Kapitel 3

VAR-Tool

3.1 UI-Mockup

3.2 Architektur

3.2.1 Geschäftsprozesse

3.2.2 Deployment

3.3 Umsetzung

3.3.1 Backend

3.3.2 Frontend

3.4 Installation

Kapitel 4

Fazit

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Quellenverzeichnis

Abkürzungsverzeichnis

bspw. beispielsweise

VM Virtuelle Maschine

u.A. unter Anderem