Vom herkömmlichen Applikationshosting Django basierter Anwendungen zur Software as a Service (SaaS) am Beispiel von OpenSlides

Die Bachelor Arbeit, die aus der beschriebenen Planung dieses Dokuments entsteht, wird bei der Intevation GmbH[2] geschrieben. Der Autor des Dokuments ist seit über 2 Jahren bei der Intevation GmbH angestellt und hat während seines Studiums dort gearbeitet. Die Intevation GmbH schreibt seit über 18 Jahren freie Software mit freier Software, weswegen im Rahmen der Bachelorarbeit ausschließlich freie und/oder Quelloffene Software betrachtet wird. Unter anderem hat der Autor an Gpg4win[3] und OpenSlides[1] mitgewirkt.

OpenSlides ist eine Django[4] basierte Web-Anwendung zur Unterstützung von Versammlungen. Mit der Software lassen sich mehrere Arbeitsabläufe abbilden, die bei der Verwaltung und Durchführung von Veranstaltungen unterstützen. OpenSlides bietet dabei unter anderem die Möglichkeit Redelisten, Tagesordnungen, Anträge und Wahlen zu leiten. Hierbei können alle beteiligten an einer Veranstaltung auf die Software zugreifen und damit in verschiedenen Rollen interagieren[1]. Intevation GmbH bietet auch Support und Hosting für OpenSlides bei Veranstaltungen an[5].

Beim herkömmlichen Applikationhosting ist aufgefallen, dass bei Veranstaltungen mit vielen Teilnehmern entsprechend viel Leistung beim Hosting der Applikation benötigen. Meist sind Veranstaltungen nur über einen beschränkten Zeitraum von wenigen Tagen. In diesen wenigen Tagen wird die gebotene Leistung gut genutzt. In den Wochen und Monaten der Vor- und Nachbereitung jedoch nicht. Hier wird jedoch die Möglichkeit gesehen durch eine Prozessvirtualisierung der Software die Leistung besser zu nutzen, um so das Angebot für den Kunden attraktiver zu gestalten und den Fußabdruck der nötigen durch geschickte Orchestrierung und Planung zu verkleinern.

1 Stand der Technik

1.1 Prozessvirtualisierung

Die Virtualisierung und die Nutzung von Software und Prozessen und die Ausgliederung von Abhängigkeitsmanagement in Container ist nicht erst ein Thema der letzten Jahre. Bereits 1979, als Unix in der Version 7 entwickelt wurde, wurde `chroot` eingeführt. Leider bietet es sich nicht für OpenSlides als Virtualisierung an, da chroot keine Root Privilegien Isolation unterstützt.[7]

Zwischen 1972 und heute sind natürlich viele neue Tools hinzugekommen, die in irgend einer Art und Weise die Virtualisierung von Prozessen unterstützen. Es sollen jedoch nur jene betrachtet werden, welche die Spezifikationen der Open Container Initiative (OCI)[6] unterstützen. Die OCI hat sich im Jahre 2015 aus verschiedenen führenden Technologieanbietern im Bereich der Container basierten Virtualisierung gegründet um einen offenen Standard zu entwickeln. Dieser wird von den Technologieführern[23] entwickelt und implementiert. Im Rahmen der Betrachtung der Technolgien zur Prozessvirtualisierung werden nur Tools betrachtet, welche diesem Standard folgen. Diese Entscheidung wurde getroffen, da abzusehen ist, dass dieser Standard noch lange existiert und das auch über verschiedene Tools hinweg. Eine Zukunft ist also gesichert.

Durch den Standard, der durch die OCI gepflegt wird, unterscheiden sich die Produkte nur noch in einigen Punkten. So kann lediglich die Beschreibungssprache zum bereitstellen eines Containers als Image unterschiedlich sein. Einmal erstellt, sind die Images jedoch durch jedes Tool nutzbar. Dann können die Images wiederum unterschiedlich durch die Werkzeuge behandelt werden. Den „Rohling“ für einen Container, nennt man Image.

Zur Zeit gibt es drei große Lösungen, welche den Richtlinien der OCI folgen. Zum einen `rkt` (rock-it gesprochen), welches von den Entwicklern des Betriebssystems “CoreOS„ im Jahre 2014 veröffentlicht wurde[8]. Des weiteren gibt es seit kurzem `railcar` von “Oracle„, welches eine auf der Programmiersprache `rust` basierende Implementation des OCI Standards ist[9]. Abschließend gibt es `docker` von der „Docker Inc.“, Initiator der OCI und mit der Veröffentlichung im Jahre 2013 seit längsten am Markt und mit über 13 Milliarden Downloads (2017) Marktführer in dem Segment[11].

1.2 Orchestrierung

Da, wie eingangs erwähnt, die Anwendung skalieren soll, wird ein Werkzeug benötigt mit dem die Anzahl der Container für die einzelnen Komponenten der Applikation angepasst werden können. Dabei wurde die Applikation in 4 Images unterteilet. Zunächst wird ein`nginx`[24]Container als „Proxy“ benötigt, um die Anwendung aus dem Web über die Standard-Ports erreichbar zu machen. Dahinter liegt `daphne` [25].`daphne` ist ein HTTP-basierter WebSocket Protokoll-Server, der über die Channel-Technolgie aus Django die `worker`-Endpunkte anspricht, die wiederum ein Container sind. Schlussendlich gibt es noch `redis`[**?**] als Container, der für den Cache bei den `worker`n sorgt und `postgres`[27] im Container als Datenbank.

Wie man sich vorstellen kann, sind der `redis` mit dem Cache und `postgres` mit den Daten, ebenso wie der Proxy und Load Balancer `nginx` schlecht zu skalieren. Diese Container dürfen also nur einmal pro Instanz der Applikation gestartet werden. In der Dokumentation zu Openslides[13] findet man den Hinweis, dass pro `daphne`-Instanz vier `worker` gestartet werden können. Diese beiden Container können also im Verhältnis $1 : 4$ skaliert werden. Das trennen anhand Virtueller Netzer mehrerer Instanzen einer Applikation und das Skalieren basiert auf Metriken oder auf „Knopfdruck“ zu machen, nennt man Orchesteriren.

Die in Kapitel 1.1 genannten Tools bringen Teilweise ihre eigenen Werkzeuge zur Orchestrierung mit, so bringt `docker` das die Engine `docker swarm`[14] mit. Diese Arbeitet nativ mit `docker` und der Docker-Engine zusammen. Zusätzlich gibt es `kubernetes`[15], welches ursprünglich von Google entwickelt wurde, mittlerweile aber in Cloud Native Computing Foundation übernommen wurde. Abschließend gibt es Tools wie `ansible`[17] oder `puppet`[16], welche eigentlich zur automatischen Software bereitstellung, Konfiguration Management und Applikations deployment hergestellt wurden, sich aber auch für die Orchestrierung von Containern (teilweise mit Plugins), einsetzen lassen.

2 Existierende Lösungen

Um mehrere Instanzen von OpenSlides auf der gleichen Hardware bereit zu stellen wurde OpenSlides Multiinstace Backend (OSMIB)[12] entwickelt. Die Instanzen sind hier zwar schon in Containern, die mit Docker gebaut werden, in der `rkt` Engine verwaltet und mit Ansible Orchestriert werden. Sie können jedoch noch nicht automatisch skalieren. Um hier mehr `worker` oder `daphne` zu starten, muss manuell in den Container geschaltet werden um die entsprechenden Änderungen vorzunehmen. Diese sind auch nach einem Neustart der Instanz verloren.

Das bestehende OSMIB bietet die Möglichkeit OpenSlides Container basierend auf verschiedenen Grund-Containern zu bereit zu stellen und sogar bestehende Instanze mit neuen Grund-Containern zu versehen. So kann man mit einem Tausch des Basis-Containers und einem Neustart der Instanz eine auf Kunden angepasste (z.B. Design) Version starten, oder die existierende Version updaten. Dazu greift das OSMIB auf eine Docker Registry[18] zu und nutzt die dort bereit gestellten Images um die Container zu updaten oder provisionieren. Des weiteren können Instanzen mit voreingestellten Administration-Benutzern und auf bestimmte Domains zeigend erstellt werden.

Diese Funktionen können über eine Weboberfläche gesteuert werden. Des weiteren bietet die Weboberfläche eine Anzeige und Steuermöglichkeit über den Status der Instanzen an. Zudem werden dort die Domains, sowie die Passwörter und Benutzernamen des Administrations-Kontos angezeigt.

3 Technologiewahl & Neuentwicklung

Durch die Spezifikationen des OCI sind viele der Technolgien untereinander Kompatibel, weswegen die Wahl für ein Tool zur Prozessvirtualisierung größtenteils unabhängig von der Wahl für ein Tool zur Orchestrierung stattfinden kann. Ledigich das Orchestrierungstool Docker Swarm ist von dem Prozessvirtualisierungstool Docker abhängig.

3.1 Prozessvirtualisierung

Wie in Kapitel 1.1 erwähnt, unterscheiden sich die Werkzeuge `rkt`, `railcar` und `docker` lediglich in der Beschreibung der Images und in der Ausführung dieser. Wenn das Image dann bereit gestellt ist, sind sie untereinander Kompatibel, können aber als breit gestellte Container wiederum unterschiedlich behandelt werden. Da `docker` der de-facto Industriestandard ist, werden die anderen Technolgien damit verglichen.

Bei einem Vergleich zwischen `docker` und `rkt` Fallen nur wenige Unterschiede auf. Einer der bestechenden Unterschiede ist jedoch, dass Container, die mit `rkt` gestartet werden mit Root-Rechten gestartet werden müssen. Dies ist bei neueren Versionen von `docker` nicht mehr nötig. Dies minimiert das Risiko, dass Container mit Sicherheitslücken Schäden im Host-System anrichten können. Des weiteren ist die Auswahl und Unterstützung an 3rd-Party Images unter `docker` deutlich größer, als bei `rkt`.[19]

Weiterhin ist die Struktur der Virtualisierung zwischen `rkt` und `docker` unterschiedlich (vgl. Abbildung 1). Wo bei `rkt` ein Prozess direkt gestartet wird, wird er bei `docker` über eine Engine verwaltet.

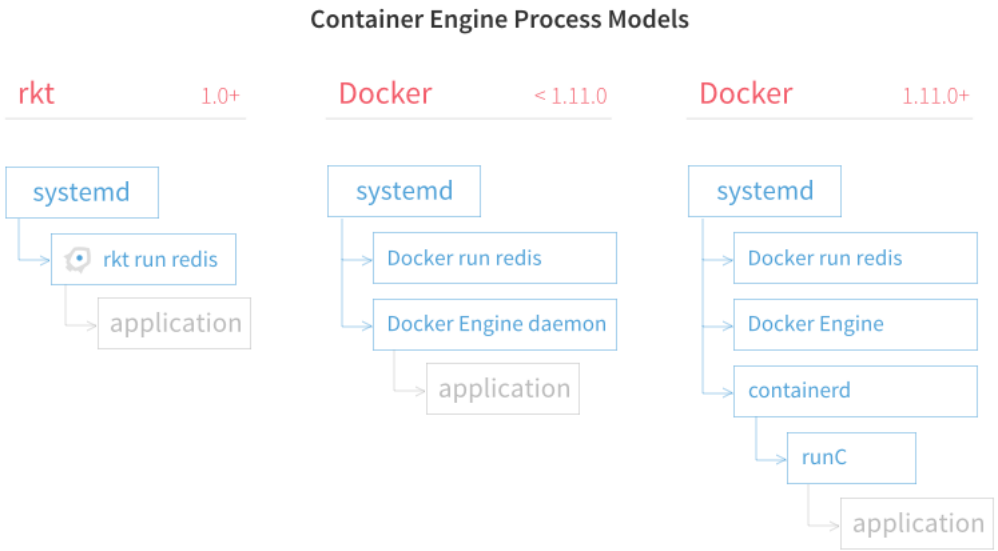

Abbildung 1: Prozessmodelle im Vergleich zwischen rkt und `docker` [20]

Die direkte Verwaltung hat den Vorteil, dass es eine Abstraktionsschicht weniger gibt. Die Engine bietet jedoch den Vorteil, dass man Containern die Möglichkeit gibt sich selbst oder sogar andere Container in der Engine zu verwalten, ohne dass man dabei zugriff auf das System geben muss. Die Abstraktionsebene kann also auch als Sicherheitslayer gesehen werden.

Ein Vergleich zwischen `docker` und `railcar` fällt schwer, da es keine wirklichen Unterschiede dazwischen gibt. Nach Aussage eine Entwicklers von `railcar` sind die Laufzeitkomponenten zu 98% identisch.[21] Des weiteren ist `railcar` nur ein drop-in für die Laufzeitkomponente von OCI basierenden Applikationen.

Die Wahl zum Tool für die Prozessvirtualisierung fällt somit auf `docker`. Dieses Tool ist bereits seit einigen Jahren im Einsatz und hierzu gibt es bereits viele Lösungen und Dokumentationen durch die Community. Des weiteren ist `docker` weiter verbreitet, weswegen die Unterstützung bei Infrastructure as a Service (IaaS) und Platform as a Service (PaaS) Anbietern einfacher ist.

3.2 Orchestrierung

Anders wie die Werkzeuge zur Prozessvirtualisierung, gibt es für die in Kapitel 1.2 genannten Tools (`docker swarm`, `ansible`, `puppet` und `kubernetes`) keinen Standard. Was jedoch schon beim lesen der Webseiten der Produkte auffällt, dass lediglich `kubernetes` direkt mit der Orchestrierung von Container-Engines wirbt.

In der aktuellen Lösung [12] werden `ansibile`-Skripte eingesetzt um die Container zu verwalten. Jedoch ist diese Lösung basiert auf einem einzelnen Container und müsste für die Orchestrierung stark erweitert werden. Die Mittel zur Container-Orchestrierung müssten hierbei selbst geschrieben werden, wie auch bei `puppet`.[17][16]

`docker swarm` bietet keine direkte Möglichkeit Orchestrierung zu betrieben. Anders wie `ansible` und `puppet` ist `docker swarm` jedoch direkt in `docker` integriert und kommuniziert somit nativ mit der Container-Engine. Jedoch müssten auch hier die Werkzeuge zur Orchestrierung selbst geschrieben werden.[14]

Unterschiedlich zu allen anderen genannten Werkzeugen bringt `kubernetes` sowohl alle Tools zur Verwaltung von Containern mit, aber auch alles um die Container zu orchestrieren. Es arbeitet eng mit der Container-Engine zusammen und man kann einfach von einem `docker swarm` dorthin migrieren, sodass auch eine lokale Entwicklung einfach fällt.[15]

Die Wahl zum Tool für die Orchestrierung fällt somit auf `docker swarm` und `kubernetes`. Auch in der Community und Wirtschaft gibt es für diese Lösung eine breite Unterstützung und `kubernetes` ist der de-facto Standard für das Orchesteriren von Applikationen. Hier bieten schon viele PaaS-Anbieter ein „`kubernetes`-as-a-Service“ an. Hier können sich allerdings auch andere Tools ergeben, je nach Dienstleister oder vom Kunden gewünschten verfahren können auch Technologien, wie z.B. CloudFoundry [22] zum Einsatz kommen.

4 Erwartetes Ergebnis

Es wird eine Lösung erwartet, die ähnlich einfach bedienbar ist wie das momentane Multiinstace Backend [12]. Man soll also Applikationen ohne Ausfallzeit updaten können. Wichtige Informationen über die Instanzen sollen schnell einsehbar sein. Die Container der Applikationen sollen agil und dynamisch auf Belastungen reagieren und skalieren.

Kern der Anforderung ist es eine Lösung zu schaffen bei der, im Gegensatz zum momentanen Zustand, die Applikationen ohne administrativen Eingriff skalieren und dass Applikationen ohne Ausfallzeit updatebar sind. Dies soll zum einen dem Kunden Ausfälle und manuell auszugleichende Belastungsengpässe ersparen, aber auch kosten in der Administration sparen.

Durch die automatische Skalierung und eine geschickte Planung von Terminen kann zudem auch die Leistung vorhandener Hardware besser genutzt werden, oder sogar in einem Pay per use (PPU) Modell angewendet werden. Dies sollte weiterhin die momentanen Kosten für einzel-Kunden senken.

Ziel ist es die momentanen Zeitfaktoren in der Administration zu messen und diese mit den Zeitfaktoren nach der Entwicklung zu vergleichen. Des weiteren sollen die Kosten zur Bereitstellung verglichen werden. Erwartet wird, dass die Zeitfaktoren, vor allem durch die geringere Einzelbetreuung für eventuelle Belastungsspitzen deutlich sinken werden und das die kosten für die Bereitstellung für den einzelnen ebenfalls sinken.

5 Planung

Datum	**Schritt**
05.2018	Anmeldung der Arbeit am Prüfungsamt
20.06.2018	Vortrag über die Arbeit im Oberseminar
01-14.07.2018	Verhindert an der Arbeit zu schreiben, da noch eine Klausur in Informatik D geschrieben werden muss für den erfolgreichen Abschluss zum Bachelor
09.2018	Abgabe der Arbeit
09.2018	Universitätspraktikum
10.2018	Abschluss mit Bachelor

Akronyme

IaaS Infrastructure as a Service

PaaS Platform as a Service

SaaS Software as a Service

OCI Open Container Initiative

OSMIB OpenSlides Multiinstace Backend

PPU Pay per use

Literatur

[1] OpenSlides.org Homepage, `https://openslides.org/`, abgerufen am 2018-03-15

[2] Intevation GmbH Homepage, `http://intevation.de/`, abgerufen am 2018-03-15

[3] Gpg4win Homepage, `https://www.gpg4win.de/`, abgerufen am 2018-03-15

[4] Django Homepage, `https://www.djangoproject.com/`, abgerufen am 2018-04-11

[5] OpenSlides.com Homepage, `https://openslides.com/`, abgerufen am 2018-03-15

[6] Open Container Initiative, `https://www.opencontainers.org/`, abgerufen am 2018-03-15

[7] „A Brief History of Containers: From 1970s chroot to Docker 2016“, von Rani Osnat, 05-05-2016 `https://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016`, abgerufen am 2018-03-15

[8] CoreOS rkt Homepage, `https://coreos.com/rkt/`, abgerufen am 2018-03-15

[9] RailCar GitHub Homepage, `https://github.com/oracle/railcar`, abgerufen am 2018-03-15

[10] Docker Homepage, `https://www.docker.com/`, abgerufen am 2018-03-15

[11] „Docker's Tools of Mass Innovation: Explosive Growth From Open-Source Containers to Commercial Platform for Modernizing and Managing Apps“, von Laura Bernheim, 2017-06-26, `http://www.hostingadvice.com/blog/dockers-explosive-growth-from-open-source-containers-to-commercial-platform/`, abgerufen am 2018-03-15

[12] OpenSlides Multiinstance Backend GitHub Page, `https://github.com/OpenSlides/openslides-multiinstance-backend`, abgerufen am 2018-03-15

[13] OpenSlides Github Page, `https://github.com/OpenSlides/OpenSlides`, abgerufen am 2018-03-15

[14] Docker Swarm Homepage, `https://docs.docker.com/engine/swarm/`, abgerufen am 2018-03-15

[15] Kubernetes Homepage, `https://kubernetes.io/`, abgerufen am 2018-03-15

[16] Puppet Homepage, `https://puppet.com/`, abgerufen am 2018-03-15

[17] Ansible Homepage, `https://www.ansible.com/`, abgerufen am 2018-03-15

[18] OpenSlides Docker Registry von emanuels, `https://hub.docker.com/r/emanuels/openslides/builds/`, abgerufen am 2018-03-15

[19] „Docker vs CoreOS Rkt“, 08-09-2017, `https://www.upguard.com/articles/docker-vs-coreos`, abgerufen am 2018-03-15

[20] „rkt vs other projects“, `https://coreos.com/rkt/docs/latest/rkt-vs-other-projects.html`, abgerufen am 2018-04-03

[21] „Oracle Releases an OCI-Based Container Runtime“, 05-07-2017 von Alex Handy, `https://thenewstack.io/oracle-opens-oci-container-runtime/`, abgerufen am 2018-04-03

[22] Cloud Foundry Website, `https://www.cloudfoundry.org/`, abgerufen am 2018-04-09

[23] OCI Member Page , `https://www.opencontainers.org/about/members`, abgerufen am 2018-04-11

[24] ngix Website, `https://nginx.org/`, abgerufen am 2018-04-11

[25] Daphne GitHub Repository, `https://github.com/django/daphne/`, abgerufen am 2018-04-11

[26] Redis Website, `https://redis.io/`, abgerufen am 2018-04-11

[27] Postgres Website, `https://www.postgresql.org/`, abgerufen am 2018-04-11