# Case Study:
# Docker Swarm Middle-Tier Betrieb in systemkritischer Hochschulumgebung einer Exzellenzuniversität

Autor: Yannick Schuberth

Contributor: Dr. Bastian Roth, Peter Brilla

Technical Reviewers: Dr. Michael Zeising

Veröffentlicht: 15. September 2020

Applies to: Multi-Tier Architecture, Docker Swarm, Load Balancing

## Zusammenfassung

Universitäten und Hochschulen haben größte Schwierigkeiten qualifiziertes Operations-Personal zu akquirieren und zu halten. Durch die Bindung an den Tarifvertrag für den öffentlichen Dienst der Länder gibt es für Hochschulen nur wenig Spielraum im Wettbewerb mit Tech-Unternehmen um die besten Köpfe im IT-Betrieb. Dies betrifft auch das Betriebsteam der Technischen Universität München (TUM), die als Vorreiter bei der Nutzung neuer Technologien insb. auch im Betrieb ständig mit dieser angespannten Personalsituation konfrontiert wird.

So ist es nur konsequent, dass die TUM mit indibit einen kompetenten Partner gefunden hat, um das Innovationstempo beibehalten zu können ohne Abstriche bei der Service-Qualität machen zu müssen. Die Aufgabenstellung war eine Umstellung und der spätere Betrieb der systemkritischen CAMPUSonline WebLogic Middletier auf Docker Swarm. Das Server-Housing sollte allerdings weiterhin vom CMIT-Team übernommen werden. Die dafür notwendigen Schnittstellen mussten technisch spezifiziert und vertraglich durch geeignete SLAs fixiert werden.

Die TUM stand mit der Entscheidung den Betrieb auszulagern unter Zugzwang, da der Software-Hersteller nach einem Technologiewechsel den Support der WebLogic-Umgebung auslaufen ließ. Die Entscheidung zu der Docker-Erweiterung Swarm fiel deshalb, da eine hochverfügbare und skalierbare Verteilung der Container auf mehrere Hosts ermöglicht werden sollte. Einzelne Services können dadurch schnell repliziert werden. Einer On-demand-Skalierung steht somit nichts mehr im Wege.

Das Team von indibit hat bereits mehrjährige Erfahrung bei der Umstellung auf Docker Swarm basierenden Systemarchitekturen. Außerdem betreibt indibit bereits Docker-Swarm-Umgebungen für 15 Universitäten und Hochschulen und konnte sich dadurch eine weitreichende Expertise aufbauen.

# 1 Ausgangslage

Die einzelnen Hosts sind für eine einfache Darstellung als farbig hinterlegte Kästen dargestellt (Abbildung 1). Die Services, die auf den jeweiligen Hosts laufen, sind grau hinterlegt.

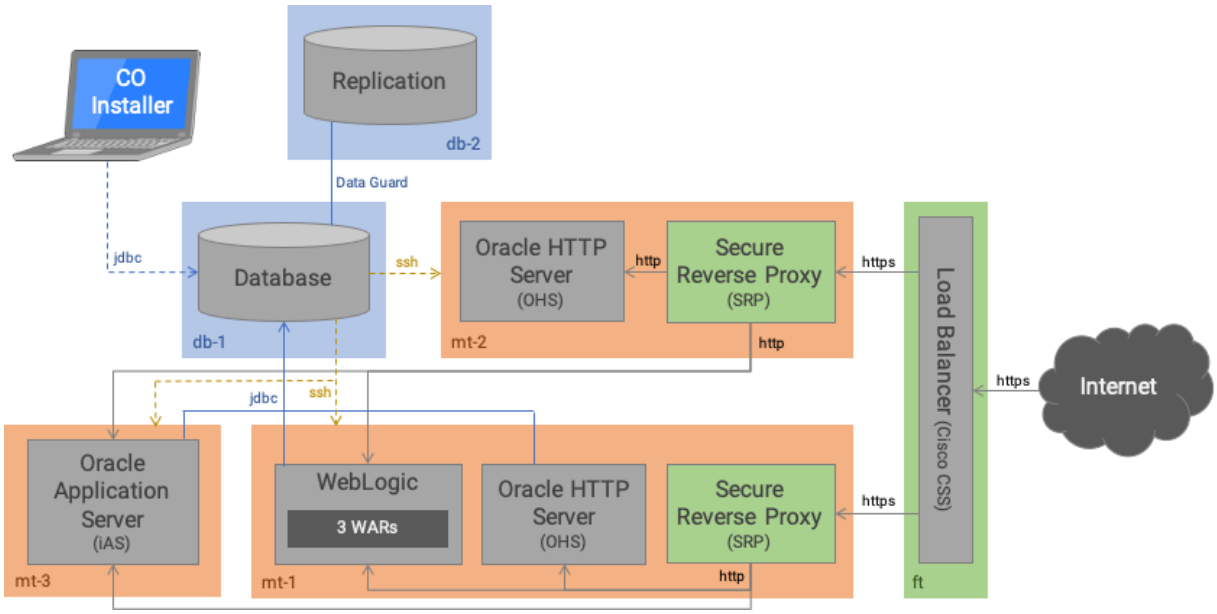


*Abbildung 1: Status quo der Systemarchitektur des Produktivsystems*

Die Systemlandschaft besteht aus einer Hauptdatenbank, die als **Database** bezeichnet wird. Sie ist per Data Guard auf die Datenbank **Replication** gespiegelt. Der **Secure Reverse Proxy (SRP)** dient zur Kapselung der dahinter liegenden Services sowie zur TLS-Verschlüsselung ins Internet über HTTP. Der **Load Balancer** dient zur Verteilung der Anfragen an die beiden dahinter liegenden **SRPs**. Sowohl **SRPs** als auch **Load Balancer** gehören aufgrund ihrer Aufgabe zum Frontend Tier und sind deshalb grün eingefärbt.

Der **Oracle Application Server (iAS)**, ist für Generierung von PDFs mittels Oracle Reports zuständig und wird mit Ende 2020 durch die vollständige Umstellung auf Apache FOP hinfällig. Zusammen mit **WebLogic** und **Oracle HTTP Server (OHS)** gehört er zum Middle Tier von CAMPUSonline. Der **OHS** dient hierbei als Bindeglied zwischen der PL/SQL-Logik in der Datenbank und dem SRP. Vereinfacht ausgedrückt ermöglicht er das Aufrufen von Stored Procedures mithilfe von Web Requests.

Im **WebLogic** laufen verschiedene JavaEE-Services als sog. Domains. Sie implementieren einige Teile der Geschäftslogik von CAMPUSonline. Langfristig soll die gesamte Geschäftslogik, die bisher in PL/SQL umgesetzt ist, nach JavaEE migriert werden.

Der **CO Installer** ist eine lokal installierte Desktopanwendung auf Java-Basis, die sich für Deployments per JDBC mit der Datenbank verbindet. Die durchgezogenen Linien bedeuten den Zugriff im regulären Betrieb, d.h. über einen Webbrowser. Die gestrichelten Linien zeigen den Zugriff im Rahmen von Deployments auf.

# 2 Zielarchitektur

Für die Umstellung auf die Zielarchitektur ergaben sich drei Umstellungsaufgaben:

1. die Trennung von Middle und Frontend Tier,
2. die Neukonzeption des Load Balancing und
3. der Wechsel von WebLogic auf Docker Swarm,

die im Folgenden genauer erläutert werden.

Für die Zielarchitektur des Produktivsystems waren mit **ft**, **mt-primary** und **mt-standby** drei neue Hosts erforderlich (Abbildung 2). Für die Testsysteme genügten zwei neue Hosts, da der Docker-Swarm nicht ausfallsicher über mehrere Hosts verteilt betrieben werden muss. Das Q-System stellt als finales Testsystem vor Umsetzung in der Produktiv-Umgebung nochmal eine exakte Replikation des Produktiv-Systems dar.

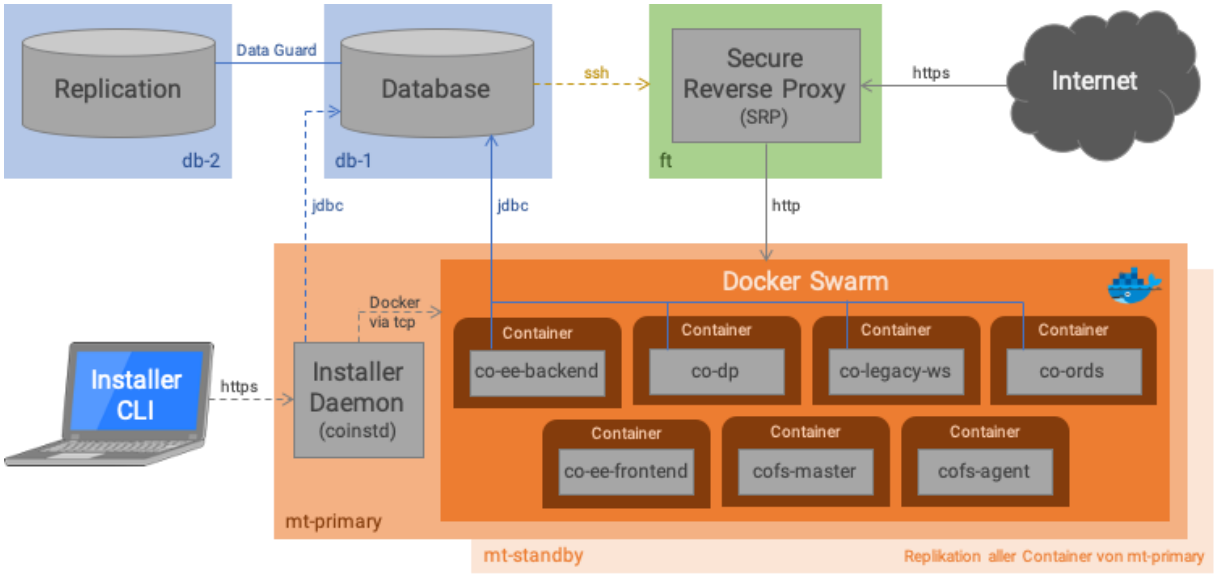


*Abbildung 2: Zielarchitektur des Produktivsystems*

Die Struktur der Datenbanken konnte dabei unverändert beibehalten werden. Das Load Balancing wurde von der Frontend Tier zur Middle Tier verschoben. Dies hatte mehrere Gründe. Zum einen besitzt Docker Swarm bereits einen integrierten Load Balancer. Dadurch konnte der Komponentenumfang reduziert und der Cisco CSS eingespart werden. Außerdem kann der Load Balancer in der Middle Tier aufgabengerichtet anhand des Ziel-Service gesteuert werden.

Mit **ft** wurde ein separater Host für **SRP** neu aufgesetzt, da SRP zuvor repliziert auf zwei Hosts des Middle Tier lief. Wir haben uns für einen weiteren Host entschieden um Wartungsschaltungen möglich zu machen, solange die Middle Tier nicht erreichbar ist, bspw. während dem Einspielen eines Systemupdates. Dadurch kannibalisieren die Services des Middle Tier nicht mehr die Ressourcen des SRP. Eine etwaig erforderliche Skalierung könnte dann durch Zuteilung weiterer Hardwareressourcen und entsprechend eingestellter Parallelisierung erfolgen.

Die Middle Tier wurde durch einen Docker Swarm mit den beiden Hosts **mt-primary** und **mt-standby** ersetzt. Dadurch konnte eine simple Hochverfügbarkeitslösung [1] umgesetzt werden, da der Swarm aus lediglich zwei Hosts besteht. Fällt mit mt-primary der sog. Swarm Leader aus, so übernimmt mt-standby die Verarbeitung eingehender Web Requests, kann aber aufgrund des Quorum-Prinzips [2] so

lange nicht mehr gesteuert werden, bis mt-primary wieder verfügbar ist. Auf mt-standby laufen repliziert alle Container, die auch auf mt-primary laufen, jedoch im sog. Hot-Standby Betrieb.

Der **WebLogic** wurde durch mehrere Container ersetzt. Als Application Server kommt in den vier Containern der ersten Zeile TomEE zum Einsatz. **cofs-master** und **cofs-agent** basieren auf Spring Boot, wohingegen **co-ee-frontend** die Dateien des Web-Clients per nginx ausliefert.

In absehbarer Zeit wird der OHS durch einen **Oracle REST Data Services Server (ORDS)** ersetzt und läuft dann ebenfalls in einem Docker Container. Beide dienen als Bindeglied zwischen der PL/SQL-Logik in der Datenbank und dem SRP. In Abbildung 2 ist der Service mit *co-ords* bereits als Container eingezeichnet.

**Installer Daemon** und **Installer CLI** lösen den ehemaligen CO Installer ab. Der **Installer Daemon** ist eine Serveranwendung, die auf dem mt-primary installiert ist, und sich um das Deployment der PL/SQL-Artefakte und der Docker Services kümmert. Das Installer CLI hingegen ist ein Kommandozeilen-werkzeug, das den Daemon steuert. Bei der TU München ist es auf einem zentralen Rechner installiert, zu dem alle Personen Zugang haben, die Updates von CAMPUSonline durchführen.

# 3 Docker-Swarm Umstellung als Enabler für die Auslagerung des Betriebs

Eine spürbare Entlastung des Betriebsteams der TUM konnte durch die alleinige Umstellung der Middle Tier allerdings noch nicht erreicht werden. Vielmehr war durch die Neukonzeption der Systemlandschaft die Voraussetzung für eine einfache Übernahme des Betriebs geschaffen.

Die Umstellung auf Docker Swarm machte es zunächst notwendig, dass indibit die gesamte Systemarchitektur sowie die zugrundeliegenden Prozesse analysiert. Die hierfür notwendige Einarbeitung erforderte eine starke Einbeziehung des Betriebsteams der TUM, da nur dort die notwendigen Spezialkenntnisse für den Betrieb vorhanden waren. Die strukturierte Aufarbeitung der langfristig gewachsenen Systemstrukturen zahlte sich aber aus. Indibit war damit gut für die Übernahme des gesamten Betriebs gerüstet. Im Ergebnis konnte das Betriebsteam der TUM spürbar entlastet werden und Service-Levels angepasst werden. Zudem wurden die technischen Kenntnisse des Betriebsteams mit den Betriebs- und Applikationskenntnissen der indibit vereint, um eine fast vollständig neue, stabilere, sicherere Netzwerk- und Systemumgebung zu schaffen, die ideal für künftige Entwicklungen vorbereitet ist.

Eine umfassende Auftragsdatenverarbeitungsvereinbarung ermöglicht den Zugriff über sichere Kanäle auf die Systeme der TUM. Dadurch können die physischen Systeme weiterhin bei der TUM verbleiben. Insgesamt konnte der Betrieb für die folgenden Tasks nach der Umstellung auf Docker Swarm vollständig durch indibit übernommen werden:

1) Quartalsweise Installation von Sicherheitspatches
2) Einspielen von Systemupdates nach Vorgaben des Systemherstellers
3) IT Service Management mit Single Point of Contact
4) Regelmäßiges Aktualisieren von CAMPUSonline (Deployment) nach Rücksprache und Koordination mit den Systemverantwortlichen der TUM

Jede Änderung an einem System des Middle und Frontend Tier (Update, Fehlerbehebung etc.) wird ITIL-konform als sog. Change Request abgebildet und dokumentiert. Außerdem überwacht indibit

kontinuierlich und proaktiv die Betriebssystemdienste, den Docker Swarm, die darin laufenden Container und den SRP.

Mit Ausnahme des Betriebs des Hypervisors (VMware) gab es keine regelmäßig wiederkehrenden Aufgaben mehr, die für das Team der TUM beim Betrieb anfallen. Fehler auf Hypervisor-Ebene treten äußerst selten auf und sind durch eine Replikation des Rechenzentrums auf zwei Standorte mit automatischem Fail-Over zusätzlich abgesichert [3]. Zudem werden sie von den im Hypervisor integrierten Monitoring-Tools üblicherweise rechtzeitig erkannt und können daher ohne Auswirkungen auf die darauf laufenden Systeme behoben werden. Deshalb wurde der Betrieb des Hypervisors im Verantwortungsbereich der TUM belassen.

Im Rahmen von regelmäßigen und spontanen Abstimmungen ist das Team der TUM maßgeblich nur noch auf organisatorischer Ebene involviert, um die zu erfüllenden Aufgaben und deren Status im Blick zu behalten, sowie die Koordination zwischen allen Beteiligten (Anwendern, Rechenzentrum, Betrieb) zu steuern. Die Arbeitslast der TUM, vor allem im Rechenzentrum, konnte wesentlich verringert werden.

Mit einem vorausschauenden Monitoring, das die Systeme rund um die Uhr überwacht, und vorsorglichen Sicherheitsmaßnahmen ist der Ausfall eines Systems außerhalb der Servicezeiten sehr ungewöhnlich. Ein frühes Einwirken auf ungewöhnliches Systemverhalten erübrigt meist die Notwendigkeit eines 24x7-Service, ohne Einbußen in Systemverfügbarkeit oder Servicequalität. Eine vorausschauende Planung der Systeme und der anfallenden Arbeiten federt außerdem noch vorhersehbare Hochlast-Zeiten ab und gewährleistet einen reibungslosen Betrieb.

Als effiziente Alternative zum Zugriff via VPN haben wir einen Jump Host eingerichtet. Dieser eigenständige Host im Netz der TUM, auf dem wichtige Hilfswerkzeuge laufen (z.B. Monitoring Proxy und screen), ermöglicht den Zugriff per ssh von festen IPs. Eine Mehrfaktor-Authentifizierung schützt zusätzlich vor fremden Zugriffen. Nach initialer Bereitstellung des Hosts durch die TUM erfolgt die Einrichtung und der weitere Betrieb vollständig durch indibit (Installation und Konfiguration der erforderlichen Tools, Monitoring, Einspielen von Applikations- und Systemupdates etc.). Die Vorteile sind die einfache Zugriffskontrolle, die für bestimmte Personen über das Hinterlegen und Entfernen von Public Keys durch indibit geregelt wird. Außerdem ist der Zugang robust gegenüber Netzstörungen von indibit oder deren Netzbetreiber, da langlaufende Prozesse nicht unterbrochen werden [4]. Des Weiteren stehen für sämtliche Operationen beim Deployment und beim Betrieb die volle Kapazität des TUM Netzes zur Verfügung. Dadurch können Zeiten, in denen die Systeme im Wartungsmodus sind, reduziert werden. Zudem stellt er als Monitoring Proxy einen gebündelten Ausgangspunkt für Überwachungsdaten dar, wodurch die Middle Tier und Frontend Tier Hosts diese nicht selbst aus dem internen Netzwerk heraus senden müssen [5].

# 4 Fazit

Outsourcing-Entscheidungen in systemkritischen Bereichen müssen mit Bedacht getroffen werden. Die TUM hat sich nach vorheriger Analyse und sorgfältiger Bewertung für die Auslagerung der Umstellung und den Betrieb der systemkritischen CAMPUSonline WebLogic Middletier auf Docker Swarm für die indibit GmbH entschieden. Für das Betriebsteam der TUM haben sich daraus eine Reihe von Vorteilen ergeben, die sich kurz- und langfristig bemerkbar machen.

Kurzfristig konnten Ressourcen frei gemacht werden, die dringend für die Wartung in anderen Operations-Bereichen sowie im Ausbau der Infrastruktur benötigt werden. Langfristig muss nun nicht mehr unbedingt mit einem internen Betrieb neuer Systeme geplant werden. Der externalisierte Betrieb ist eine ständige Option für den zukünftigen Systembetrieb und die Umstellung auf neue Architekturen.

Außerdem kann die entspanntere Personalsituation genutzt werden, um konkrete Personalengpässe in anderen Bereichen langfristig durch neue Vertretungsregelungen zu entzerren. Der ausgelagerte Bereich dient nun außerdem als Musterbeispiel für eine durchgängige Dokumentation aller Betriebs- und Wartungsprozesse. Dies wurde durch die detaillierte gemeinsame Analyse der Betriebsprozesse möglich. Dadurch kann die zukünftige Konzentration von Inselwissen vermieden werden und neues Personal schnell angelernt werden.

# Referenzen

[1] https://docs.docker.com/engine/swarm/admin_guide/#operate-manager-nodes-in-a-swarm

[2] Chanda Ray: Distributed Database Systems. 2009, Chapter 10: Distributed Recovery Management

[3] https://www.vmware.com/de/products/vsphere/replication.html

[4] https://www.heise.de/ix/artikel/Lebensversicherung-506716.html

[5] https://www.zabbix.com/documentation/current/manual/distributed_monitoring/proxies