# Gruppenarbeit 3 - Abschliessende Analyse des Providers und Cloud Application Design

Alexander van Schie & Oli Dias

May 20, 2019

# Contents

# 1 Analyse Service Level Agreegment (SLA)

- 1. Es wird eine etwas andere Art von SLO's gemacht, nämlich wird das Problem nach Schwerheitsgrad klassifiziert und je schlimmer es ist, desto schneller muss RedHat reagieren (https://access.redhat.com/support/offerings/openshift/sla)
- 2.
- 3. Generell werden keine Metriken oder Messwerte erwähnt. Vielmehr werden Probleme zusammengefasst und nach Schwerheitsgrad klassifiziert. (https://www.openshift.com/legal/terms/)
- 4. Je nach SLA müssen verschieden Dinge eingehalten werden. Was mir persönlich als wichtig erscheint ist eine Vereinbarung bezüglich dem Kundensupport innerhalb einer gewissen Zeit da (a) ein Unterbruch meiner Applikation je nachdem grosse Konsequenzen für mein Unternehmen haben kann. Grundsätzlich sollte ein Cloud-Provider ausgewählt werden, der quasi to big to fail ist (b).
- 5. Je nach Branch muss man sich mit den Datenschutzbestimmungen des Cloud Providers auseinander setzen. Beispielsweise wäre es für Banken nicht gerade ideal, Kundendaten auf ausländische Server zu migrieren/verwalten

# 2 Hands-On Lakeside Mutual

Die zu deployenden Komponenten lassen sich in zwei Kategorien einteilen; Frontend und Backend. Das Setup der beiden Teile des Backends werden im nachfolgenden Unterkapitel besprochen, das Frontend im Unterkapitel 2.2.

## 2.1 Spring Boot Backend Setup

Für das Migrieren des Backends der Lakeside Mutual DDD Applikation eignet sich für die beiden Komponenten Customer-Core und Customer-Management-Backend ein Wildfly Projekt auf Openshift. Dies sollte analog wie schon beim Deployment der Self-Information Applikation funktionieren, da alle drei Applikationen Spring Boot verwenden.

Dazu sind folgende Schritte notwendig:

1. Hinzufügen der `.s2i/bin/run` Datei in beiden Spring Boot Apps, die den Befehl für das Ausführen des gebuildeten Jar Files enthält:

```
1 exec java -jar /wildfly/standalone/deployments/
2  customer-management-backend-0.0.1-SNAPSHOT.jar
```

2. Anpassen der Application-Properties in Customer-Management-Backend:
   - Property `customercore.baseURL` auf die URL des Customer Cores einstellen. Diese ist sichtbar gemäss Abbildung 1 in Openshift Cloud herauszufinden.
   - Property `server.port` aus derselben Übersicht auslesen und setzen (Default 8080)
3. Anpassen der Application-Properties in Customer-Core:
   - Property `server.port` setzen (Default 8080)

Da Customer-Management-Backend von der Komponente Customer-Core abhängig ist, muss Customer-Core zuerst deployt werden. Ansonsten wirft die Applikation eine Laufzeit-Exception, da keine Verbindung zum Core hergestellt werden kann.

## 2.2 Frontend Deployment

Openshift bietet Node.js Projekte an, welches sich für die React App eignet.

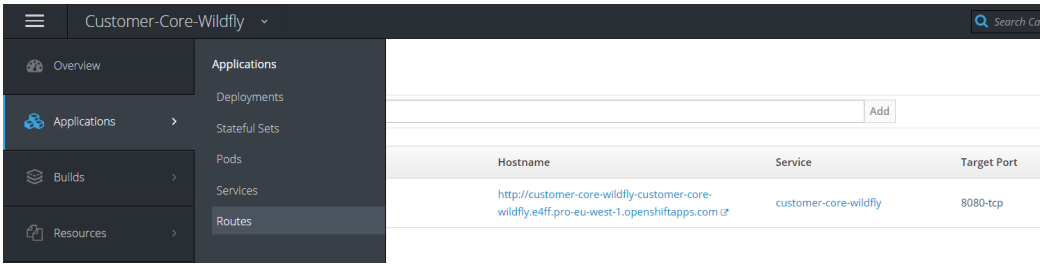


Figure 1: Navigation zu Routes

Nachdem das Nodejs-Projekt erstellt und das GitHub-Repository erstellt ist, beginnt ein zuständiger Pod gleich mit dem Build. Unglücklicherweise wurde mit unserem Setup an dieser Stelle der Build unterbrochen aufgrund von Limiten, die unsere Subscription setzt. Da das Frontend aber von beiden anderen Komponenten abhängig ist, bleibt uns nichts anderes übrig, als die Frontend-Komponente lokal auszuführen.

| Name | Last Build | Status | Duration | Created | Type | Source |
|---|---|---|---|---|---|---|
| customer-frontend | #6 | Out of memory killed | 2 minutes, 32 seconds | 14 days ago | Source | https://github.com/olidias/Cloud_customer-management-frontend.git |

Figure 2: Build-Failure aufgrund von Memory-Limiten

Für die lokale Ausführung wird Nodejs benötigt. Ist dies installiert, wird noch eine zusätzliche Konfiguration nötig: Wir müssen auf das Management-Backend zugreifen können und dafür die URL anpassen. Wie im vorgehenden Kapitel schon erwähnt, findet man den URL des Management-Backends in dessen Projekt auf Openshift unter Applications → Routes. Die URL muss im File `src/config.js` angepasst werden:

```
export const customerManagementBackend =
 getEnvironmentVariable(
  "REACT_APP_CUSTOMER_MANAGEMENT_BACKEND",
  "http://customer-management-backend.e4ff.
    pro-eu-west-1.openshiftapps.com/"
)
```

### 2.3 Herausforderungen

# 3 Hands-On Persistence

# 4 Twelve-Factor Apps

| Engineering Projekt (Waitless) | |
|---|---|
| Erfüllt | Nicht erfüllt |
| Codebase | Disposability |
| Dependencies | Dev/prod parity |
| Config | Logs |
| Backing services | Admin processes |
| Build, release, run | |
| Processes | |
| Port binding | |
| Concurrency | |
| Port binding | |

## 4.1 Anpassungen

- Disposability:
- Dev/prod parity: Die ganze Applikation wurde nur auf einem System entwickelt. Sobald Code eingecheckt und die Tests erfolgreich waren, war der Code bereits produktiv.
- Logs: Das Verhalten der Applikation wurde mit Enduser-Tests überprüft. Mit der Implementation von Logs werden alle Informationen festgehalten.
- Admin processes:

# 5 Security Features und Assessment

In den Terms und Conditions von Openshift ist ganz klar zu entnehmen, dass der Anwedender für Themen des Datenschutzes die Verantwortung übernimmt. So hat der Anwender sicherzustellen, dass die Daten der Anwender seiner Applikation geschützt werden. Dies beinhaltet die Implementation von Datnschutzrichtlinien, welche rechtlich abgestimmt sind. Zudem müssen die Anwender darüber informiert werden, dass ihre Daten auf der Infrstruktur von Red Hat abgelegt wird und sie dem somit zustimmen.

Red Hat gibt bekannt, dass folgende Technologien von ihrem PaaS unterstützt werden:

- SELinux
- Process, network, and storage separation
- Stateful and stateless inspection firewall
- Proactive monitoring of capacity limits
- Intrusion detection
- Port monitoring
- Pam namespace
- Security compliance frameworks
- RPM verification and vulnerabilities updated
- Remote logging
- Encrypted communications

https://www.openshift.com/policy/security/

## 5.1 Checkliste Security Assessment

1. Encrypted Communication/ (Security) -> 10 Openshift nutzt Kubernetes und die Kommunikation bei Kubernetes ist standardmässig mit TLS verschlüsselt.

2. Backups -> 3 Backups müssen selbst gemacht werden, es gibt hierfür keinen Service. Positive ist jedoch, dass das Vorgehen dokumentiert ist.

3. Location -> 5 Openshift wird vermutlich Data Centers an mehreren geographischen Standorten haben, kommuniziert dies jedoch nicht öffentlich.

4. Server Redundancy (Data Centre Physical Security, Disaster Tolerance) -> 8 Da Openshift mit Kubernetes arbeitet, besteht auch die Möglichkeit der Nutzung von Replicas, diese müssen jedoch selbst definiert werden.

5. Service Authentification (for Administration stuff) -> 10 Authentifizierungserfahren laufen über OAuth und bietet alle gängigen Möglichkeiten (`https://docs.openshift.com/enterprise/3.0/admin_guide/configuring_authentication.html`)

6. Appropriate SLA (with Incident Handling) -> 7 Es gibt ein SLA, in dem wichtige Punkte zur Einhaltung der Servicebedingungen festgehalten sind. Was jedoch nicht zu entnehmen ist, sind Strafen, die bei Nichteinhaltung fällig werden.

7. Data Isolation (Customer based, Data Integrity) Kubernetes NetworkPolicy wird voll unterstützt und Projekte können ebenfalls in einer isolierten Umgebung betrieben werden.

8. Transparency (Monitoring) -> 8 Quotas sowie laufende Prozesse können eingesehen werden

9. Network design und Logging -> 5 Standardmässig werden Logs geschrieben, diese waren unserer Erfahrungen nach unzureichend (was das Fehlerhandling erschwerte)

# 6 SWOT-Assessment von Cloud Provider und Cloud Offering

| | | Interne Analyse | |
|---|---|---|---|
| | | Stärken | Schwächen |
| Externe Analyse | Chancen | ▪ Web App Design<br>▪ Einfache Projektinitialisierung<br>▪ Command Line Interface (CLI) | ▪ Limitierte Ressourcen<br>▪ Limitierete Projektauswahl<br>▪ Limitierte Abonnemente<br>▪ Preis<br>▪ Web App Struktur ? |
| | Risiken | ▪ Nutzung von Kubernetes<br>▪ Ausbaubare Service Dokumentation<br>▪ Chance zur Änderung (Nicht to big to change) | ▪ Transparenz Fehlerbehandlung<br>▪ Transparenz Nutzungsbedingungen<br>▪ Ablehnung jeglicher Sicherheitsaspekte<br>▪ Kleine Community |

S indicates standardization ongoing activities

| Kriterium | Erfüllt | Kommentar |
|---|---|---|
| Umfangreicher Technologiekatalog | Jein | Welches Angebot stellt uns der Anbieter zur Verfügu |
| Unternehmungsgrösse des Anbieters | Nein | Grosse Unternehmen haben tendenziel mehr Erfahrur |
| Customizing | Jein | Ist es möglich, Ressourcen nach beliebigem Bedarf zu |
| Dokumentation | Ja | |
| Community | Nein | Gibt es eine Community, die bei häufigen Problemen |

# 7 Provider Evaluation Checkliste

# 8 Management Summary