

Ex 03: Analyse Swisscom & Cloud Application Design

Cloud Solutions

**Abteilung Informatik
Hochschule für Technik Rapperswil**

Autoren: Andreas Stalder, David Meister
Datum: 29. Mai 2017

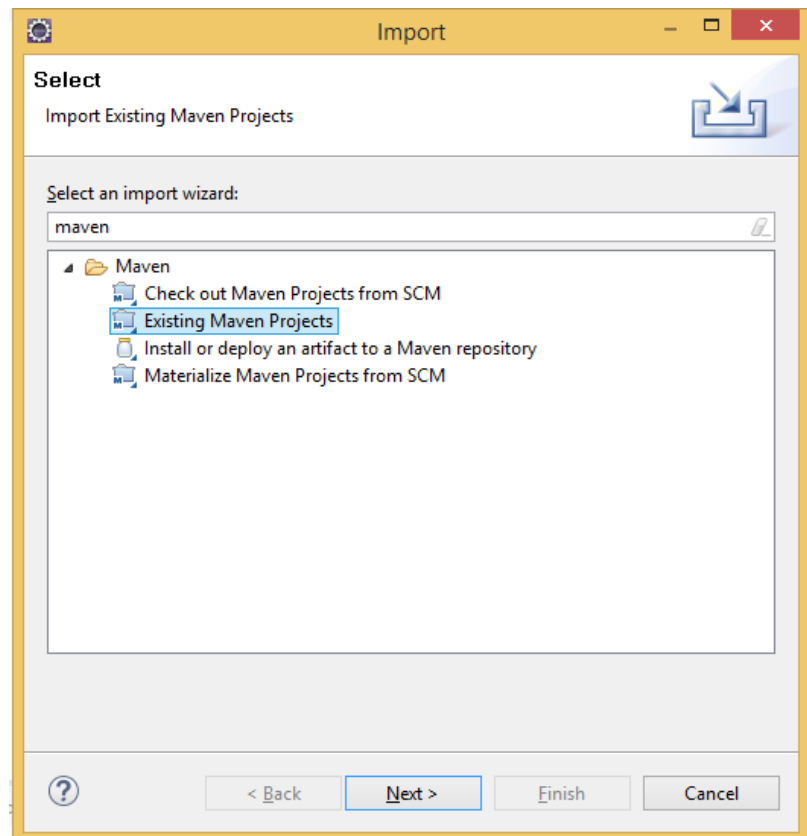
Inhaltsverzeichnis

1	Hands-On: DDD Sample Application	3
1.1	Import der DDD Applikation in die IDE	3
1.2	Deployment in die Swisscom PaaS Cloud	6
1.3	Fazit	11
2	Hands-On: Erweiterung um relationale Datenbank	12
2.1	Security Überlegungen	16
2.2	Fazit	16
2.3	Weitere Storage Offerings	16
3	Hands-On: NoSQL-Persistence in der Cloud (am Beispiel MongoDB)	17
3.1	Anleitung	17
3.2	Fragen	19
4	Analyse: Service Level Agreements(SLAs)	20
4.1	Fragen	20
5	Konzept: Cloud Computing Patterns (CCP)	22
5.1	Abbildung auf das CCP Architecture Overview Diagram	22
5.2	Cloud-Deployment Modellierung	23
6	Analyse: SWOT-Assessment von Cloud Provider und Cloud Offering	24
7	Konzept: Provider Evaluation Checkliste	25
8	Analyse: Management Summary	26
	Abbildungsverzeichnis	27

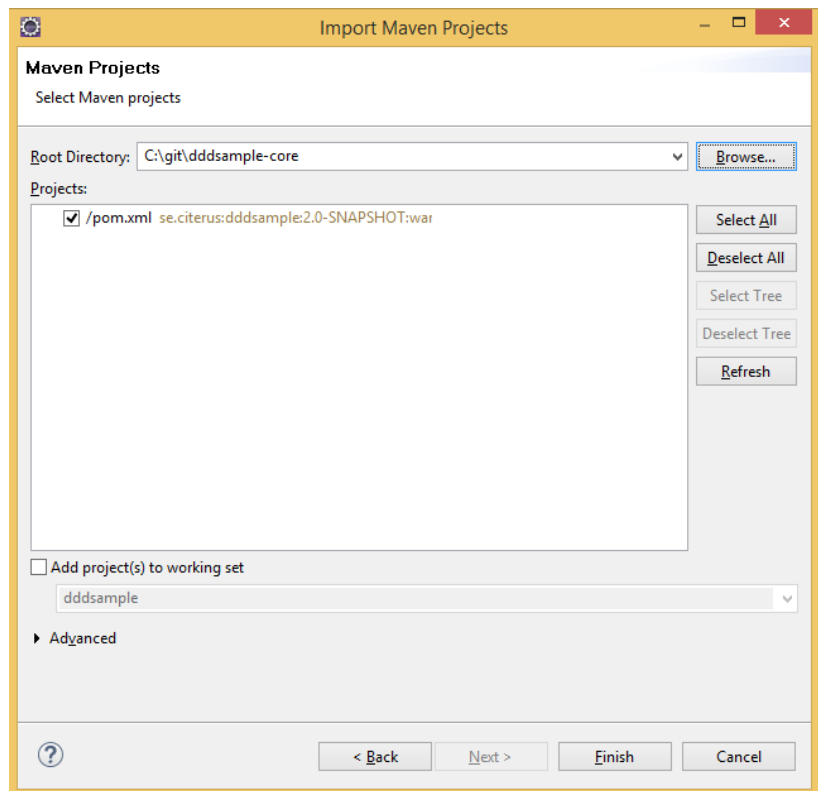
1 Hands-On: DDD Sample Application

1.1 Import der DDD Applikation in die IDE

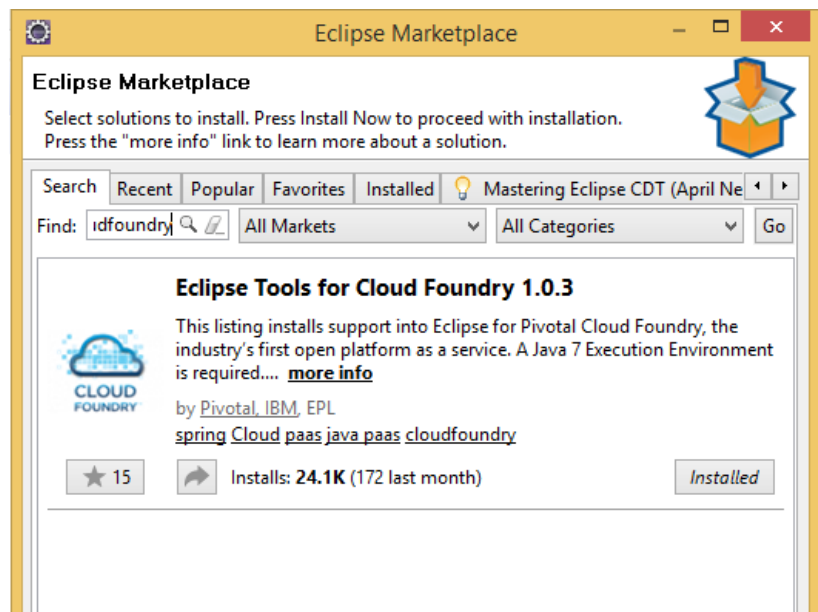
Die Vorlage Applikation kann in Eclipse als bestehendes Maven Projekt importiert werden.



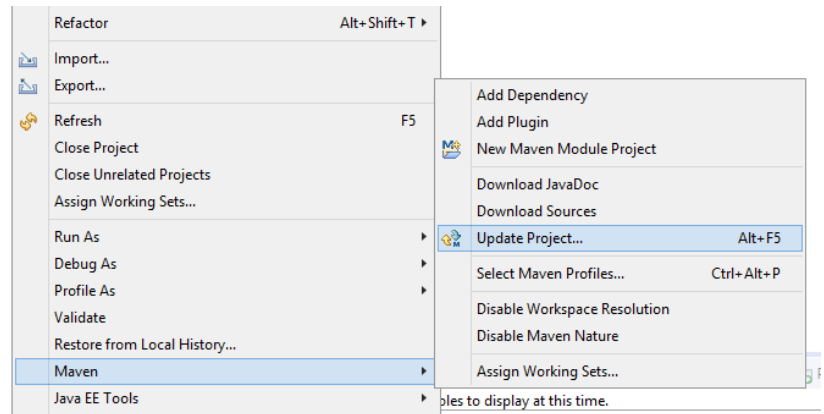
Im Projektverzeichnis die pom.xml Datei auswählen.



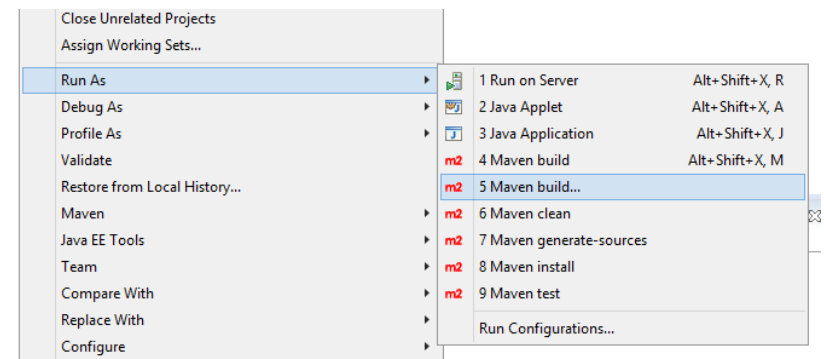
Die Swisscom PaaS Cloud arbeitet mit Cloud Foundry. Das Cloud Foundry Plugin für Eclipse kann über den Marketplace installiert werden.



Nun müssen die Maven Dependencies geladen werden.

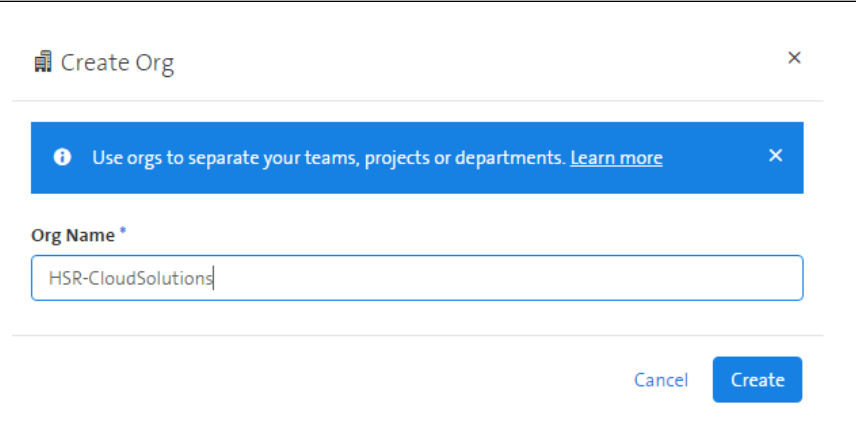
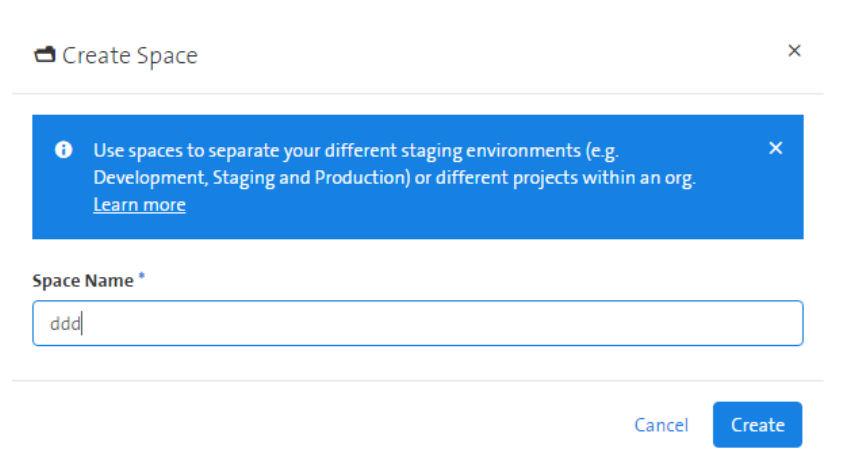


Jetzt kann der Build-Prozess gestartet werden.

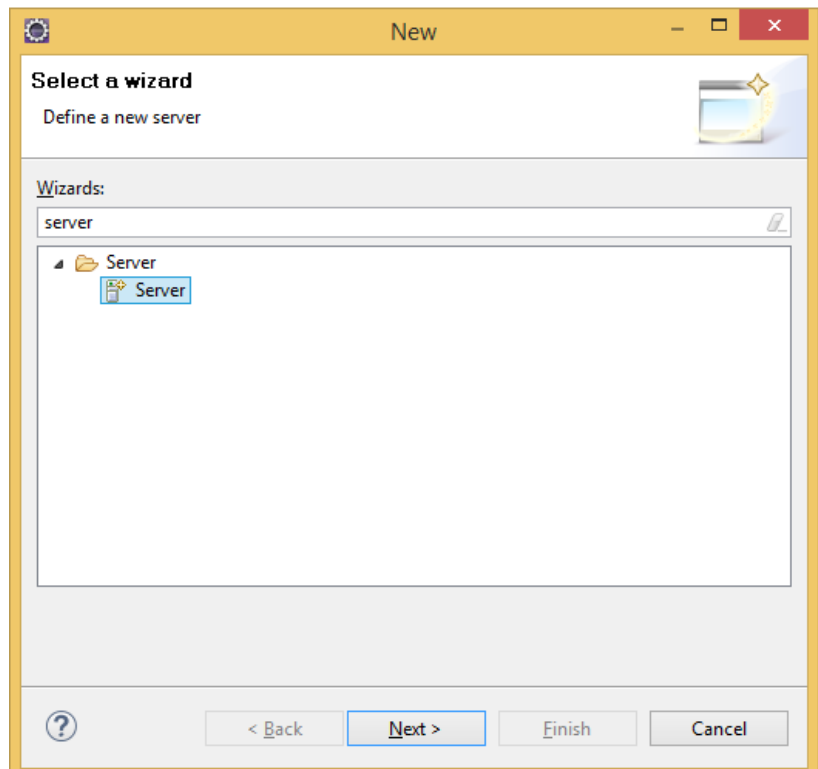


1.2 Deployment in die Swisscom PaaS Cloud

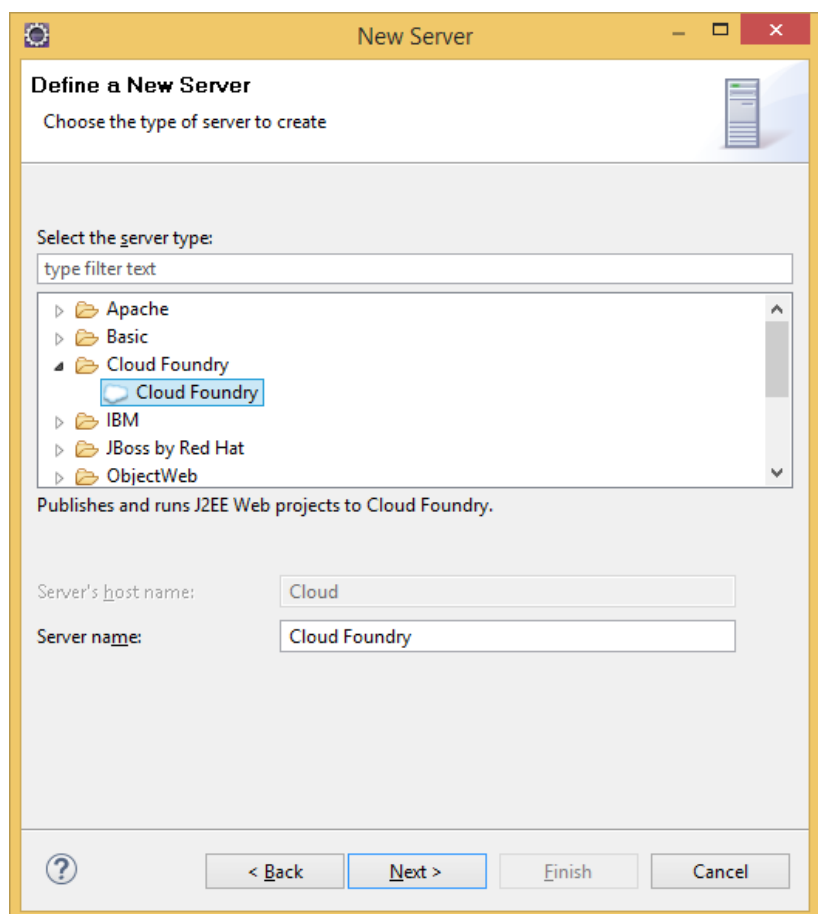
Bevor die Applikation in die Cloud gepusht werden kann muss ein Space angelegt werden. Danach kann die ddd-App in diesen Space deployed werden.

<p>Eine Organisation kann mehrere Spaces enthalten. Eine Applikation wird in einen Space deployed. Wir erstellen deshalb eine neue Org.</p>	
<p>Als Space Name wählen wir ddd (Name der App).</p>	

In Eclipse muss ein neuer Server erfasst werden. Anstatt Tomcat verwenden wir nun einen Cloud Foundry Server



Durch das zuvor im Marketplace installierte Plugin kann nun ein Cloud Foundry Server ausgewählt werden.



Die Logindaten des Swisscom Application Cloud Accounts müssen hier hinterlegt werden. Die URL „https://api.lyra-836.appcloud.swisscom.com“ muss erfasst werden.

The screenshot shows a 'New Server' window with a yellow title bar. Inside, the 'Cloud Foundry Account' section is active. It contains a sub-section 'Account Information' with a checkbox for 'Use a one-time password to login (SSO)'. Below this are input fields for 'Email' (dmeister@hsr.ch) and 'Password' (masked with dots). A 'URL' label is followed by a dropdown menu showing 'Swisscom - https://api.lyra-836.appcloud.swisscom.com' and a 'Manage Cloud...' button. At the bottom of the form are three buttons: 'Validate Account', 'Register Account...', and 'Sign Up'. The bottom of the window features a navigation bar with a help icon, '< Back', 'Next >', 'Finish' (highlighted in blue), and 'Cancel'.

New Server

Cloud Foundry Account

Press 'Validate Account', 'Next', 'Finish' to validate credentials.

Account Information

☐ Use a one-time password to login (SSO)

Email: dmeister@hsr.ch

Password: ●●●●●●●●

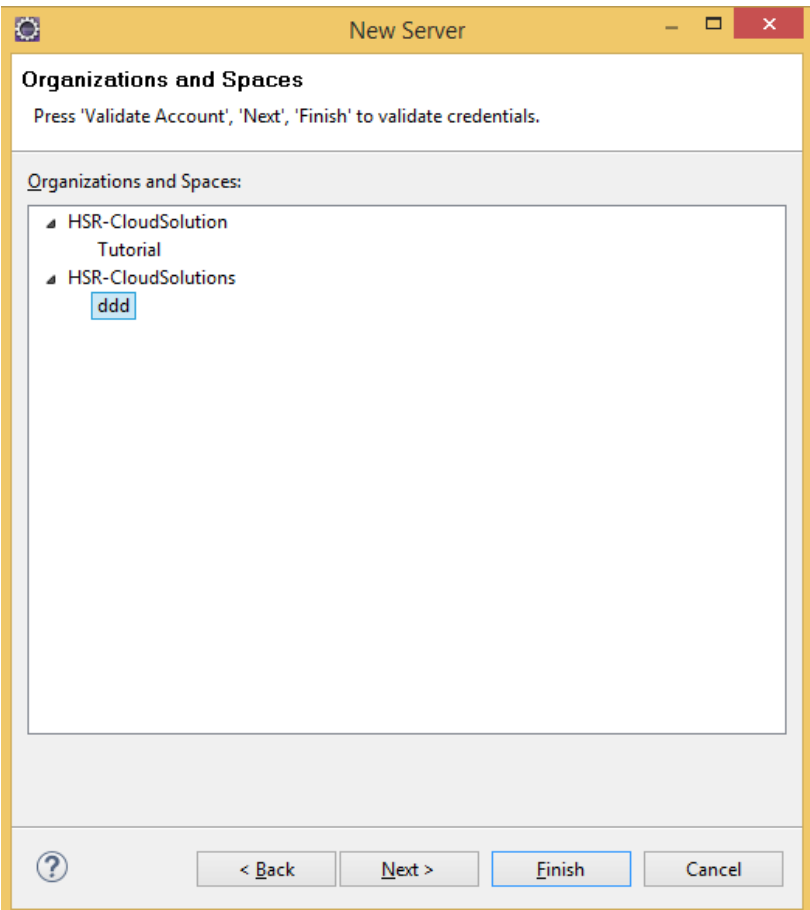
URL:

Swisscom - https://api.lyra-836.appcloud.swisscom.com Manage Cloud...

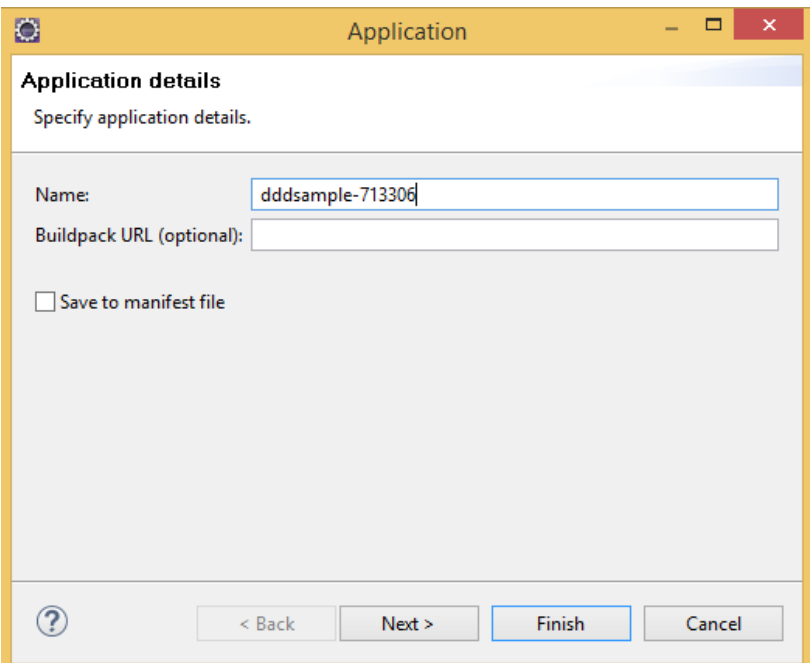
Validate Account Register Account... Sign Up

? < Back Next > Finish Cancel

Die zuvor erstellte Org und Space sollten nun erscheinen. Den gewünschten Space auswählen um fortzufahren.



Den App Namen kann man belassen oder falls gewünscht anpassen.



Falls benötigt kann das Memory Limit erhöht werden.

Application

Launch deployment

Specify the deployment details

Subdomain:

Domain: ▼

Deployed URL:

Memory Limit (MB):

☒ Start application on deployment

Nun startet der Deployment Prozess, nach kurzer Zeit ist er erfolgreich abgeschlossen.

Publishing to Cloud Foundry...

Publishing to Cloud Foundry...

Getting stats - dddsample-713306

☐ Always run in background


Markers Properties Servers Data Source Explorer Snippets Console

Cloud Foundry--HSR-CloudSolutions--ddd--dddsample-713306

```
Checking application - dddsample-713306
Generating application archive - dddsample-713306
Pushing application - dddsample-713306
Creating application - dddsample-713306
Application successfully pushed
Starting application - dddsample-713306
[Application Running Check] - Checking if application is running - dddsample-713306. Please wait...
```

Auf der Application Cloud Console erscheint nun die laufende App.

HSR-CloudSolu... / d... / dddsample-71... Estimated Monthly Cost: CHF 12.00

 dddsample-713306 Started

Instances - 1 + Memory Limit - 512 MB +

Disk Limit 1 GB

Restart Restage Stop Delete

Instances Events Logs Env Upload Routes

Services

#	STATUS	SINCE	CPU	MEMORY	DISK
0	Running	a minute	0.3%	358.4 MB	160.2 MB

1.3 Fazit

Der grundlegende Teil dieser Aufgabe war mit der Swisscom Application Cloud sehr angenehm zu lösen. Durch die Online-Dokumentation wird man gut durch das Deployment durchgeführt. Die Applikation selbst, respektive der Code musste nicht angepasst werden.


Anfangs hatten wir der Ausführung von gewissen Funktionen. Es stellte sich schnell heraus, dass wir vergessen haben, die Maven Dependencies zu laden und das Projekt zu builden.

2 Hands-On: Erweiterung um relationale Datenbank

Die DDD Sample App arbeitet standardmässig mit HyperSQL DB (HSQLDB). Dies ist in der Datei `jdbc.properties` unter `/src/main/resources` ersichtlich.

Die Swisscom Application Cloud bietet keinen MySQL Service. Stattdessen wird MariaDB als vertreter relationaler Datenbank angeboten.

MariaDB muss auf dem zuvor erstellten Space (ddd) hinzugefügt werden. Für diese Aufgabe reicht die kleinste Variante mit 1GB Storage und 10 Connections.

 Create MariaDB ×

Service Name *


Service Plan *

1GB Storage Capacity 10 Connections High Availability Small	8GB Storage Capacity 15 Connections High Availability Medium	16GB Storage Capacity 100 Connections High Availability Large
---	---	--

Estimated Monthly Cost **CHF 6.00**

Sobald erstellt, müssen im Reiter „Service Keys“ noch Zugriffsschlüssel erstellt werden. Diese müssen dann in der verwendeten App zur Authentifizierung hinterlegt werden.

HSR-CloudSolutions / ddd / RelationalDB Estimated Monthly Cost: CHF 6.00

 RelationalDB Started

[How to manage](#) [Delete](#)


General Events Backups **Service Keys** Apps

SERVICE KEYS ? [+ Create Service Key](#)

No service keys yet...

Den Keys kann ein beliebiger Name hinterlegt werden.

HSR-CloudSolutions / ddd / RelationalDB Estimated Monthly Cost: CHF 6.00

 RelationalDB Started

[How to manage](#) [Delete](#)

General Events Backups **Service Keys** Apps

SERVICE KEYS ?

[X Cancel](#) [✓ Save](#)

No service keys yet...

Die benötigten Informationen liegen nun im JSON Format da. Die private IPv4 Adresse im host-Feld verrät, dass ein externer Zugriff ausserhalb der Swisscom Application Cloud nicht möglich ist. Sowohl Username und Passwort sind ein zufällig generierter String.

ddd-keys

```
{
  "host": "10.0.20.18",
  "hostname": "10.0.20.18",
  "port": 3306,
  "name": "CF_2035A39F_EB1D_4CB7_BAB7_21DC96F998BA",
  "database": "CF_2035A39F_EB1D_4CB7_BAB7_21DC96F998BA",
  "username": "7XmyQy8z2SimelYN",
  "password": "fz02XBkTccGyhCAD",
  "database_uri": "mysql://7XmyQy8z2SimelYN:fz02XBkTccGyhCAD@10.0.20.18:3306/CF_2035A39F_EB1D_4CB7_BAB7_21DC96F998BA?reconnect=true",
  "uri": "mysql://7XmyQy8z2SimelYN:fz02XBkTccGyhCAD@10.0.20.18:3306/CF_2035A39F_EB1D_4CB7_BAB7_21DC96F998BA?reconnect=true",
  "jdbcUrl": "jdbc:mysql://10.0.20.18:3306/CF_2035A39F_EB1D_4CB7_BAB7_21DC96F998BA?user=7XmyQy8z2SimelYN&password=fz02XBkTccGyhCAD"
}
```

Der bestehenden App kann nun der zuvor erstellte DB Service gebunden werden.

HSR-CloudSolu... / d... / dddsample-71... Estimated Monthly Cost: CHF 12.00



dddsample-713306

Started

Instances − 1 +

Memory Limit − 512 MB +

Disk Limit 1 GB

Restart

Restage

Stop

Delete

Instances

Events

Logs

Env

Upload

Routes

Services

BOUND SERVICES ?

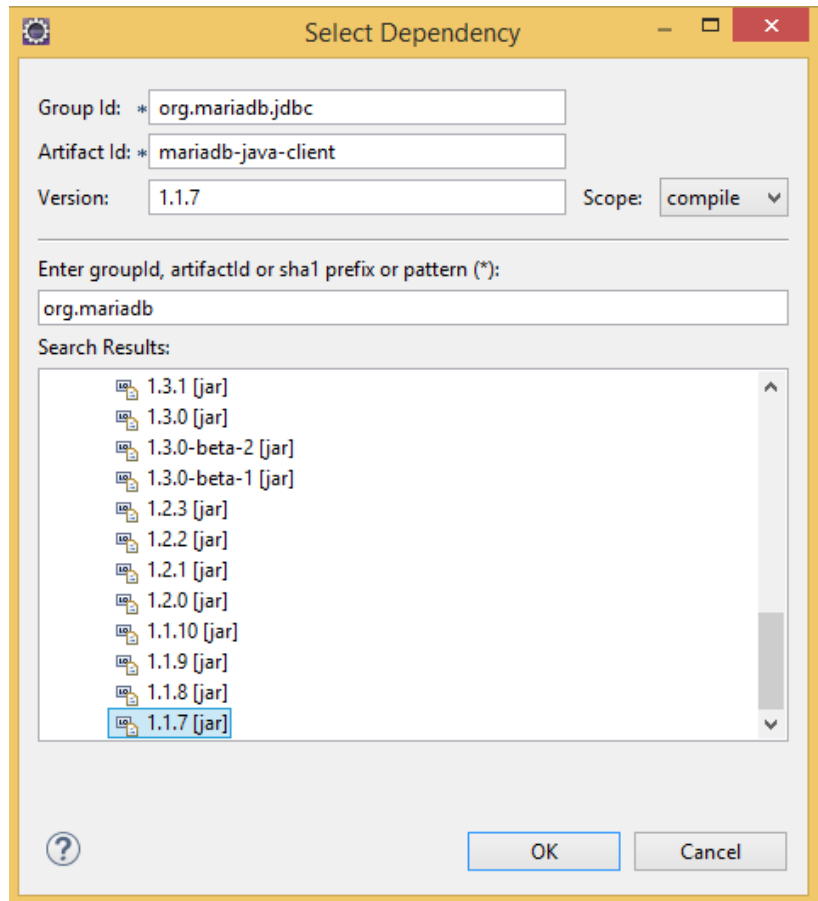
+ Bind Service



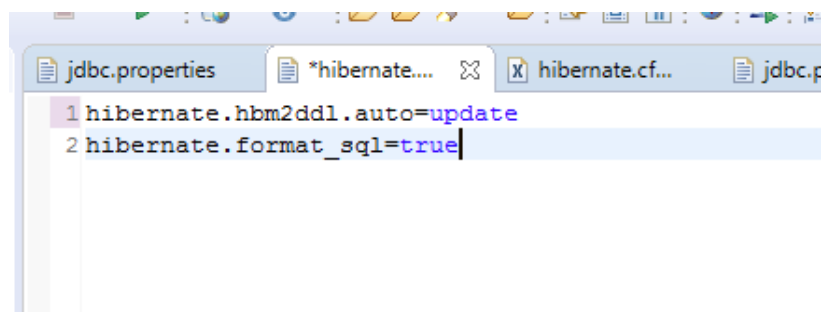
RelationalID.1 bound app

Started

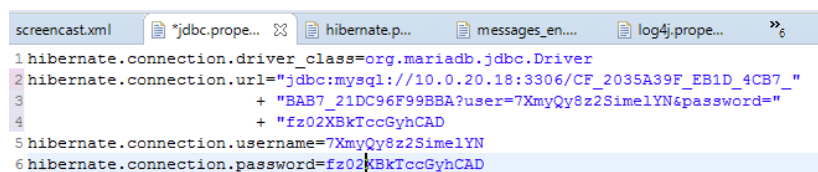
Wir benötigen für die App nun die MariaDB Dependencies, welche wir einfach mit Maven hinzufügen können. Wir verwenden die als funktionierend empfohlene Version 1.1.7.



Damit die Änderungen in der Datenbank auch nach dem Neustart persistent sind, müssen wir eine Anpassung der Konfiguration im File hibernate.properties vornehmen. Im Feld hibernate.hbm2ddl.auto müssen wir die Einstellung von „create drop“(nicht persistent) nach „update“ändern.



Nun muss nur noch im File jdbc.properties der zu verwendende Treiber und der Connection String angepasst werden.



Die Änderungen müssen nun mit Maven Update, resp. Maven Build angepasst und neu gepusht werden.

2.1 Security Überlegungen

Die Datenbankservices sind bei Swisscom nur über IP Adressen aus dem privaten Adressbereich erreichbar. Dies lässt schliessen, dass Swisscom selbst in ihrem Datacenter die Datenbanken hostet. Eine direkte Anbindung der Datenbank ausserhalb der Swisscom App Cloud ist daher nicht möglich.

Die Zugriffe zur Datenbank sind mittels zufallsgenerierten Strings für Username und Passworte authentisiert. Eine TLS verschlüsselte Verbindung zwischen App und Datenbank wäre ebenfalls wünschenswert und im Falle eines Zugriffs aus dem Internet (hier nicht der Fall) zwingend notwendig. Wir haben leider keine Informationen betreffend Verschlüsselung der Verbindungen gefunden, und ein Sniffen des Netzwerkverkehrs ist für uns leider nicht ohne weiteres Möglich.

2.2 Fazit

Die Anbindung der MariaDB ist sehr einfach. In der Applikation muss nur der zu verwendende Connection String angepasst werden. Das Erstellen des Services und der Keys ist einfach und scheint gelungen. Einzig, dass keine Informationen betreffend Verschlüsselung vorhanden sind, ist nicht optimal.

Wir hatten keine gröberen Probleme, haben aber festgestellt, dass nach dem Starten der App noch etwas gewartet werden muss, ansonsten kann die App abstürzen. Dies liegt wohl an längeren Ladezeiten im Hintergrund.

Die Dokumentation musste nicht konsultiert werden. Die Erstellung des Services ist sehr einfach und benötigt keine Erklärungen. In der DDD App mussten wir die zu verändernden Files kurz suchen, im Grossen und Ganzen hatten wir aber wenig Probleme.

2.3 Weitere Storage Offerings

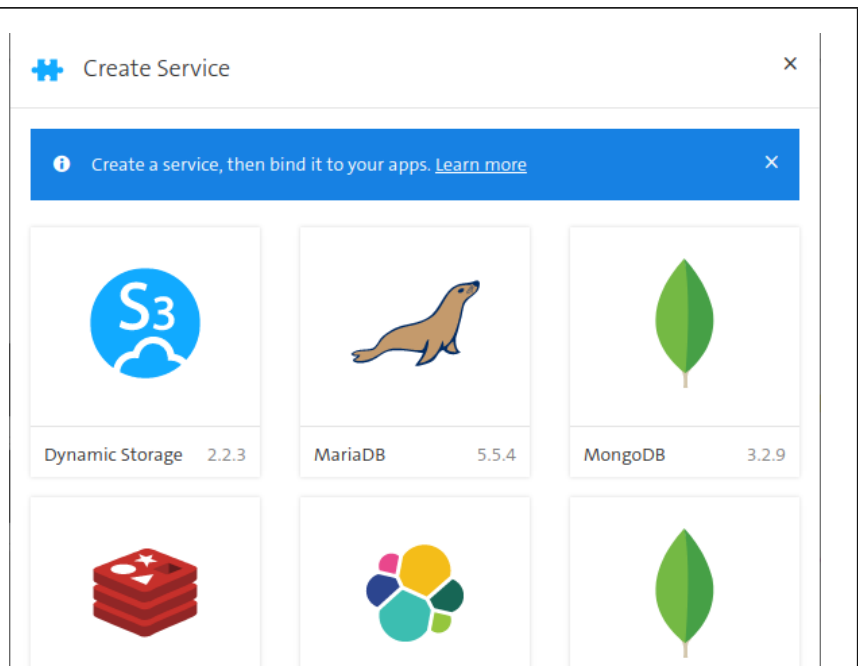
Mit MariaDB und MongoDB werden die Storage Offerings „Relational Database“ und „Key-Value Storage“ gemäss Cloud Computing Patterns (CCP) angeboten. Als in Memory-Store, welcher ebenfalls dem CCP „Key-Value Storage“ zugeordnet werden kann, wird Redis angeboten. Als Speicher für grosse Datasets bietet Swisscom den „S3 Dynamic Storage“. Dieser Service ist dem CCP „Blob Storage“ zuzuordnen.

3 Hands-On: NoSQL-Persistence in der Cloud (am Beispiel MongoDB)

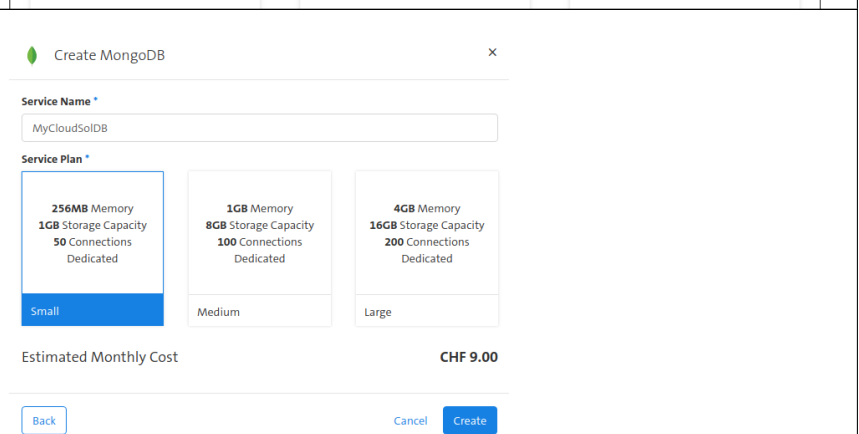
3.1 Anleitung

Für diese Aufgabe wurde das ContractManagement verwendet. Dieses bereitete aber keine grossen Probleme.

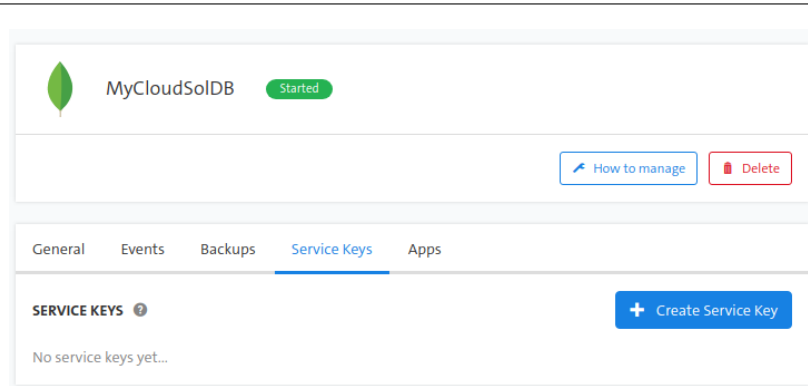
Auf der PaaS Plattform der Swisscom kann man sehr einfach einen neuen Service hinzufügen. Dazu wählt man nur Create Service und klickt auf MongoDB.



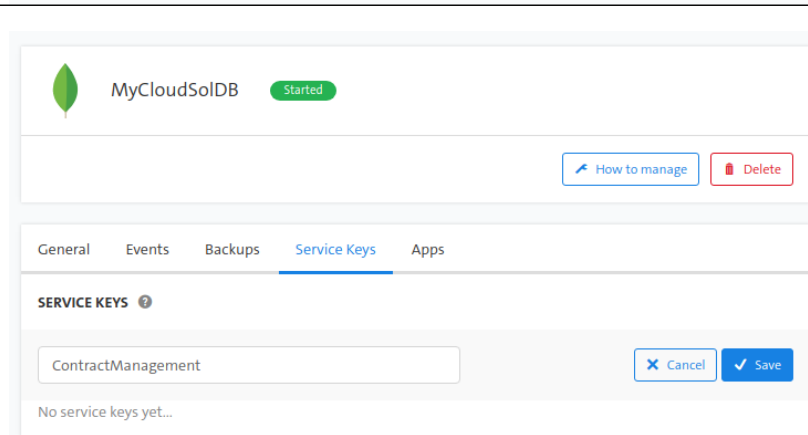
Nun vergibt man einen Namen und wählt die gewünschte Grösse. Für dieses kleine Projekt reicht Small vollkommen aus.



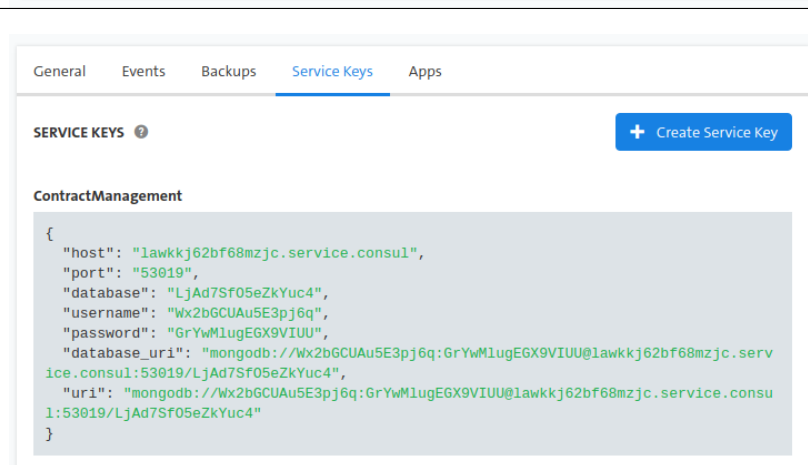
Dies erstellt automatisch die neue MongoDB. Nach ca. 1-2 Minuten ist diese bereit und kann konfiguriert werden. Damit man eine Verbindung zur Datenbank herstellen kann, muss man einen Service Key erstellen. Dies geschieht durch einen klick auf Create Service Key.



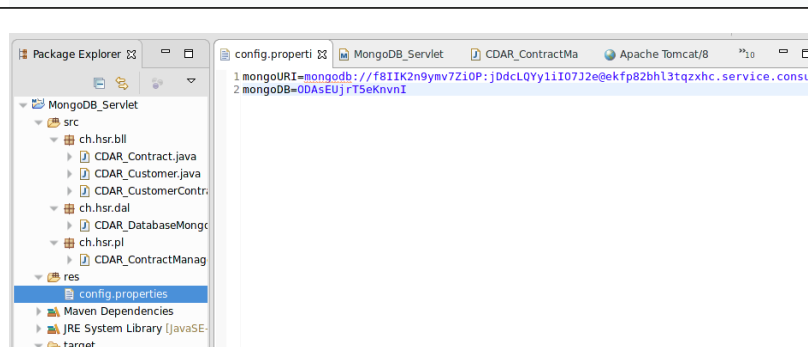
Für den Service Key gibt man einen passenden Namen und klickt auf Save.



Nun erhält man die Zugangsdaten und kann diese kopieren...



... und in der Applikation hinterlegen. Nun funktioniert die Verbindung zur MongoDB und man muss die Applikation wie bei Aufgabe 1 auf die Plattform laden.



3.2 Fragen

3.2.1 Was hat wie erwartete funktioniert?

Eigentlich hat alles wie geplant funktioniert. Einzig bei Eclipse hatten wir zuerst Probleme, da wir es zuerst Local mit Tomcat starten wollten, was aber nicht funktionierte. Doch mit dem Hochladen hat die Applikation sofort funktioniert. Grössere Konfigurationen mussten nicht gemacht werden.

3.2.2 Was hat nicht funktioniert?

Nichts. Alles hat ohne Probleme funktioniert.

3.2.3 Wie schätzen Sie die Sicherheit dieses Datenbank-Hostings ein?

Die Datenbank ist wie bei der MariaDB Variante nur intern erreichbar. Als Zugangspunkt erhält man nur den MongoDB Hostnamen, Username und das Passwort. Daher ist diese Variante auch sehr wahrscheinlich bei der Swisscom intern gehostet.

Alle Verbindungsdaten erhält man beim Erstellen der Datenbank. Es wird ein Username sowie ein Passwort generiert. Dazu erhält man den Host und den Port der MongoDB. Das Passwort und der Benutzername kann nicht separat geändert werden. Um eine neue Kombination zu erhalten, muss ein neuer Service Key erstellt werden. Durch diese Daten kann die Applikation nun auf die Datenbank zugreifen. Wie auch bei der MariaDB haben wir bei der MongoDB keine Informationen zur Verschlüsselung.

4 Analyse: Service Level Agreements(SLAs)

4.1 Fragen

4.1.1 Gibt es ein oder mehrere Service Level Objectives (SLOs)?

Bei der Swisscom gibt es mehrere Service Level Objectives. Es wird in verschiedene Kategorien unterteilt, wie zum Beispiel Availability, Security oder Continuity. Dabei wird jeder Punkt nochmals in drei SLA Stufen unterteilt. Diese sind Standard Public, Standard Virtual Private und Premium Virtual Private. So sind alle SLOs sehr genau beschrieben. Leider bietet die Swisscom momentan aber nur Best Effort an. Die SLAs sind für spätere Releases geplant, stehen aber schon bereit. Genauere angaben, ab wann diese gelten, konnten nicht gefunden werden.

4.1.2 Questions to Ask

How are uptime and availability calculated?

Diese Angabe ist bei der Swisscom vorhanden und wird wie folgt beschrieben:

The availability of each of the marketplace cloud services are measured by transactions on a reference service instance on the production. If these measurements show, that service is running, the availability status is set to „available” for this service. Unless otherwise stated, availability of the reference service is checked every 60 seconds and if two subsequent checks are failing the service is considered as unavailable.

Dabei wird nicht zwischen Uptime und Availability unterschieden.

What should be in place for me to be covered?

Die Swisscom unterscheidet zwischen "Normal Availability" und "High Availability". Normal Availability bedeutet das man nur eine Instanz gestartet hat. High Availability bedeutet ein Cluster von Nodes. Das heisst mindestens zwei Nodes müssen gestartet sein. Zwischen Normal und High Availability liegen je nach Bezahlmodel 0.3 oder 0.4 Prozent Verfügbarkeit.

What about performance degradation as opposed to hard downtime?

Es werden zwei verschiedene SLAs angegeben. Der erste ist für die Instanz, sowie für das Cloud Management. Der zweite ist für den Marketplace Cloud Services. Beide enthalten unterschiedliche SLOs, welche wieder in die verschiedenen Bezahlmodelle unterteilt sind.

What are the penalties for SLA violations?

Zu diesem Bereich gibt es keine genaueren Angaben. Im Dokument ist dazu nichts zu finden und man muss sich wahrscheinlich bei der Swisscom melden. Ob und wie man ausgezahlt wird, war auch nicht auffindbar.

What do I have to do to request a credit?

Wie bereits in der vorherigen Frage geschrieben, findet man zu diesem Thema nicht wirklich eine Angabe.

4.1.3 Fehlen bestimmte Evaluationskriterien?

Bei dem White Paper von Dimension Data geht es hauptsächlich um Uptime und Availability. Heutzutage sollte man auch immer an das Thema Security denken. Dies wird viel zu häufig vernachlässigt. Der Rest deckt aber sehr viel ab und ist gut als Evaluationshilfe geeignet.

4.1.4 Ist die Einhaltung und Existenz von SLAs wichtig?

- a) Bei der Entscheidung ob man sein Produkt in die Cloud verlagert, sind die SLAs enorm wichtig. Diese vergleicht man meistens mit seinen eigenen Möglichkeiten. Zum Beispiel die Verfügbarkeit kann man dies einfach vergleichen. Möchte man eine hochverfügbare Umgebung aufbauen, oder ist eine fremde Cloud kostengünstiger. Dabei verlässt man sich immer auf die SLAs und möchte auch mit dieser rechnen können.
- b) Auch bei der Wahl des Providers sind die SLAs enorm wichtig. Es gibt immer wie mehr Cloudanbieter und ohne SLAs kann man diese kaum miteinander vergleichen. Auch das einhalten ist wichtig, besonders für den Anbieter. Denn sobald die SLAs mehrmals gebrochen wurden, hat der Anbieter mit Kundenverlust zu rechnen.

4.1.5 Weitere wichtige Dokumente

[SLA-Definitions - Enterprise Customers](#)
[Besondere Bedienungen](#)

5 Konzept: Cloud Computing Patterns (CCP)

Bei der Swisscom wird für die PaaS-Umgebung die Lösung von Cloud Foundry eingesetzt. Die Architektur sieht folgendermassen aus:

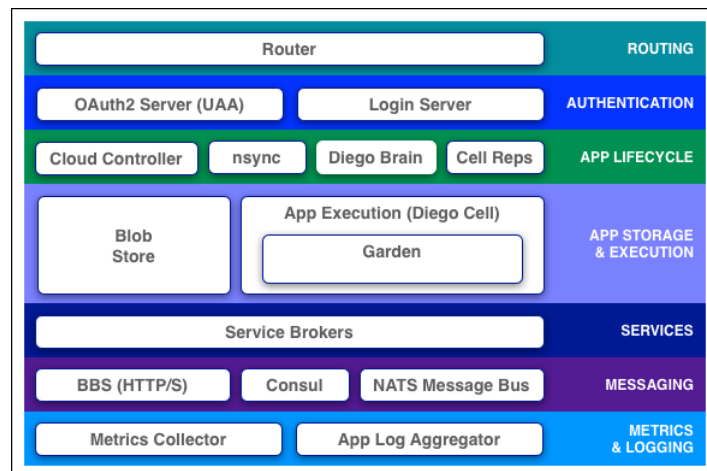


Abbildung 5.1: Cloud Foundry Architektur

Zum Vergleich ist hier nochmals das Standardarchitektur Pattern:

5.1 Abbildung auf das CCP Architecture Overview Diagram

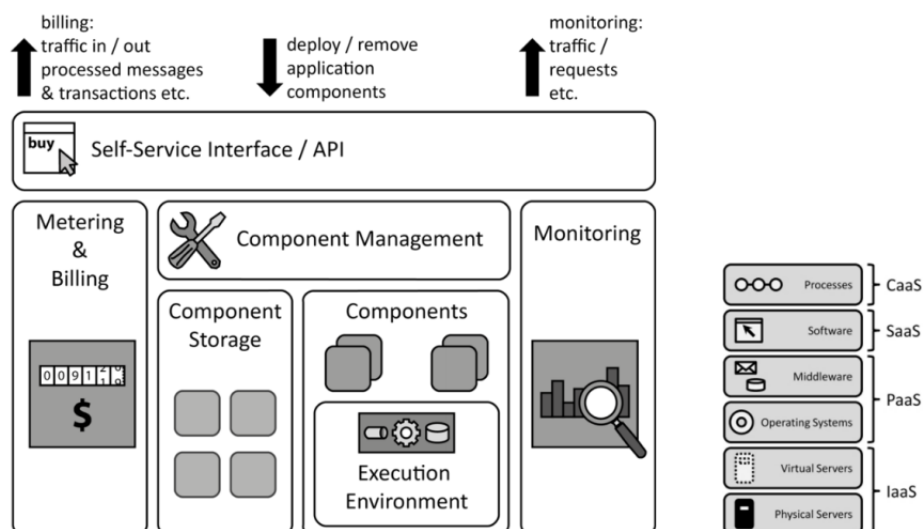


Abbildung 5.2: CCP Architektur Diagramm

5.1.1 Mappingtabelle

CCP Architecture Overview Diagram	Swisscom - Cloud Foundry
Self-Service Interface / API	Router, Login Server / UAA
Metering und Billing	Metrics Collector, App Log Aggregator
Component Management	Cloud Controller
Monitoring	Health Manager
Component Storage	Blob Store
Components	Warden
Execution Enviroment	Application Execution (DEA)

5.2 Cloud-Deployment Modellierung

Für die Modellierungsaufgabe haben wir unsere akutelle Bachelorarbeit genommen, welche wir momentan am Entwickeln sind. Diese Applikation ist ein skalierbares IoT-Management Tool.

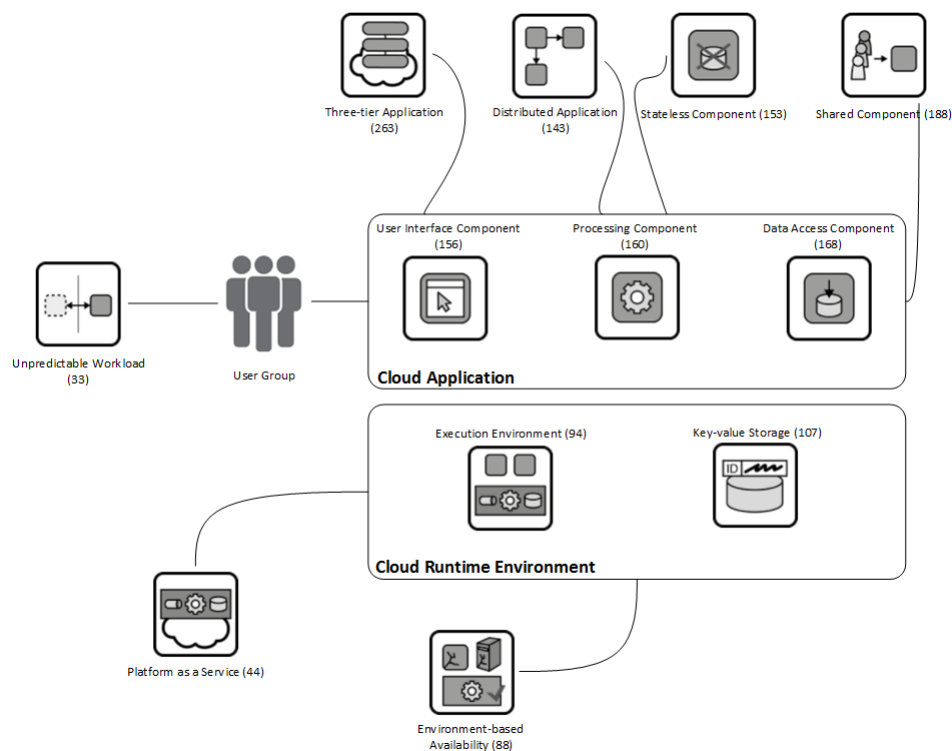


Abbildung 5.3: Cloud-Deployment Modellierung der Bachelorarbeit

5.2.1 Fragen

Ist eine Decomposition-Strategie erkennbar?

Es ist das Distributed Application Pattern erkennbar. Die Applikation wurde auf verschiedene Layer aufgeteilt. Es gibt einen Presentaion, Service, und Storage-Layer. All diese haben eine unabhängige Funktio-nalität und sind separat austauschbar.

Wir würden Sie die Applikation refactoren?

Die Applikation müsste noch besser voneinander getrennt werden. Alle Komponenten sollte loser gestaltet sein, damit jeder Teil separat überwacht und skaliert werden könnte. So würde die Applikation noch viel mehr von einer Cloud-Umgebung profitieren. Der Einsatz von Load-Balancer wäre so durchaus möglich. Oder auch andere Operation Management Pattern könnten viel einfacher umgesetzt werden.

Welches Workload Pattern liegt vor?

Bei der Applikation ist das Unpredictable Workload Pattern erkennbar. Da es sich um ein Managementsystem handelt, kann man nicht vorhersagen, wann ein Benutzer die Applikation nutzt. Würde es fixe Wartungs- und Managementfenster geben, könnte die Last vorhergesagt werden. Aber sonst ist dies sehr schwer. Man weiss nur, dass die Last zu Arbeitszeiten sicher viel höher ist, denn um diese Zeit wird die Applikation eingesetzt.

Wie sind Application State Management und Session State Management gelöst?

Das Application State Management wird nach dem Pattern Stateless Component umgesetzt. Alle Anfragen werden ohne State an die Applikation gesendet und werden verarbeitet. Es wurde explizit auf einen State verzichtet.

Beim Session State Management haben wir nur ein Login vorgesehen. Es wird jedes Mal gefragt ob der Benutzer eingeloggt ist. Weitere Session States sind in dieser Applikation nicht geplant.

Welche Cloud Computing Patterns aus der Rubrik Cloud Application Components sehen Sie?

Aus dieser Kategorie werden die Pattern Stateless Component, User Interface Component, Processing Component, Data Access Component. Diese Pattern reichen, damit die in Aufgabe 1 erwähnte Decomposition-Strategie umgesetzt werden kann. Weitere Pattern wären von Vorteil, aber würden zu einer grossen Umstellung der Applikation führen. Beim Weiterentwickeln der Applikation muss dies beachtet werden, damit die Applikation noch besser für eine PaaS-Umgebung taugt.

Sollte Ihrer Meinung nach das Watchdog Pattern implementiert werden?

Nein, dies sollte nicht implementiert werden. Bevor dies möglich wäre, müsste man die Applikation zuerst noch umbauen. Die einzelnen Komponenten sind zu sehr miteinander verbunden, um sie separat überwachen zu können. Einzig der LwM2M Server könnte man überwachen und bei Fehlern sofort wieder Neustarten, da dieser separat von dem Rest der Applikation läuft. Aber auch für dies muss der Server noch umgebaut werden, damit die Applikation die Anfragen in einer Queue zwischenspeichern kann, damit beim Starten des neuen Servers keine Probleme entstehen.

6 Analyse: SWOT-Assessment von Cloud Provider und Cloud Offering

	Positiv (Nützlich)	Negativ (Schädlich)
Interne	Stärken <ul style="list-style-type: none"> • Schweizer Datenschutzgesetz • Grosse Produktvielfalt • Markenansetzen in der Schweiz • Grosse Community mit Cloud Foundry • Einfaches, gut verständliches Pricing 	Schwächen <ul style="list-style-type: none"> • Viele Ausfälle • Teurer als Konkurrenten • Kein Auto-Scaling der Applikationen • Globale Erreichbarkeit
Externe	Chancen <ul style="list-style-type: none"> • Ausbau im europäischen Markt (Schweizer Datenschutzgesetze) • Unterstützung durch Open Source Community (Cloud Foundry) • Wachstum in der Schweiz dank Kundennähe 	Bedrohungen (Risiken) <ul style="list-style-type: none"> • Kundenverluste durch Ausfälle (Negativwerbung) • Konkurrenz durch grosse, marktführende Anbieter

7 Konzept: Provider Evaluation Checkliste

	Erfüllt?	Kommentar
Kriterium 1	Ja	•
Kriterium 2	Nein	•
Kriterium 3	•	•
Kriterium 4	•	•
Kriterium 5	•	•
Kriterium 6	•	•
Kriterium 7	•	•
Kriterium 8	•	•
Kriterium 9	•	•
Kriterium 10	•	•

8 Analyse: Management Summary

Maximal 1 Seite

Abbildungsverzeichnis

5.1	Cloud Foundry Architektur	22
5.2	CCP Architektur Diagramm	22