

Cloud Fundamentals

Cloud Solutions FS 2017

Abteilung Informatik
Hochschule für Technik Rapperswil

Autoren: Andreas Stalder, David Meister
Datum: 19. März 2017

Inhaltsverzeichnis

1	Hello World	3
1.1	Providerwahl	3
1.2	Anleitung	3
2	OSSM-Definition	10
3	Cloud Computing Patterns	11
3.1	Process Offering	11
3.2	Workload	11
3.3	Komponenten	12
3.4	Übersichtsgrafik	13
4	Self-Information	14
5	Preisrecherche	15
6	Preisvergleich Hosting vs IaaS vs PaaS	16
7	Twelve-Factor Apps	17
	Abbildungsverzeichnis	18

1 Hello World

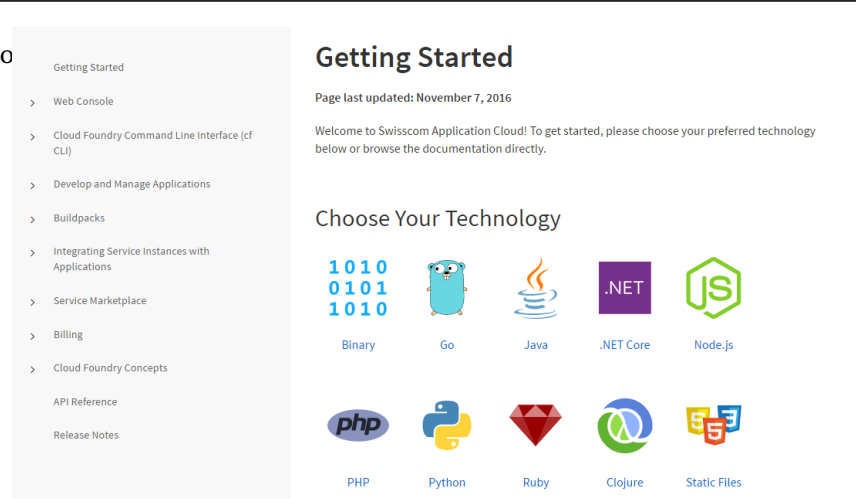
1.1 Providerwahl

Wir haben uns für Swisscom als Provider entschieden, da es uns wichtig war, einen lokalen, heimischen Anbieter bezüglich Cloud Computing besser kennenzulernen. Swisscom ist in vielen Bereichen der IT und Telekommunikation schweizweit marktführend.

In einem Gastvortrag an der HSR im Herbst 2016 ist ein Vertreter der Swisscom erschienen und hat von den Vorzügen und der gelebten Swissness erzählt. Da wir noch nie mit PaaS Produkten im Allgemeinen gearbeitet haben war dies die perfekte Möglichkeit um in die Thematik reinzufinden.

1.2 Anleitung

Der Einstieg fiel allgemein gesehen leicht. Swisscom verwendet „Cloud Foundry“ für ihre Application Cloud. Mittels Tutorials auf der Webseite wird man für den Einstieg gut „an der Hand genommen“.

<p>Unter https://docs.developer.swisscom.com/getting-started/ findet man Tutorials für die verschiedenen PaaS Produkte wie z.B. Python, Java, Node.js</p>	
<p>Als allererstes benötigt man das Cloud Foundry Command Line Interface. Auf der Webseite sind Beschreibungen für die Installation unter Linux/Macintosh/Windows zu finden. (Windows in dieser Anleitung)</p>	<h3>Windows Installation</h3> <p>To use the cf CLI installer for Windows, perform the following steps:</p> <ol style="list-style-type: none">1. Download the Windows installer.2. Unpack the zip file.3. Double click the <code>cf CLI</code> executable.4. When prompted, click Install, then Close.5. To verify your installation, open a terminal window and type <code>cf</code>. If your installation was successful, the cf CLI help listing appears.

Nach erfolgreicher Installation ist das „cf“-Tool per Windows Commandline (cmd) benutzbar

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Master Dave>cf
cf version 6.25.0+787326d.2017-02-28, Cloud Foundry command line tool
Usage: cf [global options] command [arguments...] [command options]

Before getting started:
  config login,l target,t
  help,h logout,lo

Application lifecycle:
  apps,a run-task,rt events
  push,p logs set-env,se
  start,st ssh create-app-manifest
  stop,sp app
  restart,rs env,e
  restage,rg scale

Services integration:
  marketplace,m create-user-provided-service,cups
  services,s update-user-provided-service,uups
  create-service,cs create-service-key,csk
  update-service,ds delete-service-key,dsk
  delete-service,ds service-keys,sk
  service service-key
  bind-service,bs bind-route-service,brs
  unbind-service,us unbind-route-service,urs

Route and domain management:
  routes,r delete-route create-domain
  domains map-route
  create-route unmap-route

Space management:
  spaces create-space set-space-role
  space-users delete-space unset-space-role

Org management:
  orgs,o set-org-role
  org-users unset-org-role

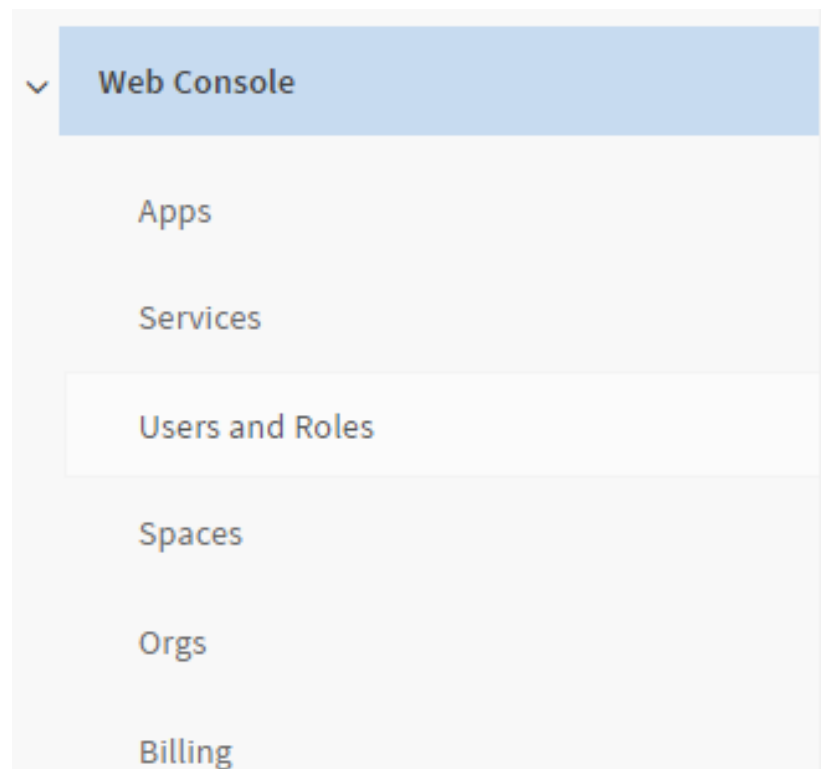
CLI plugin management:
  plugins add-plugin-repo repo-plugins
  install-plugin list-plugin-repos

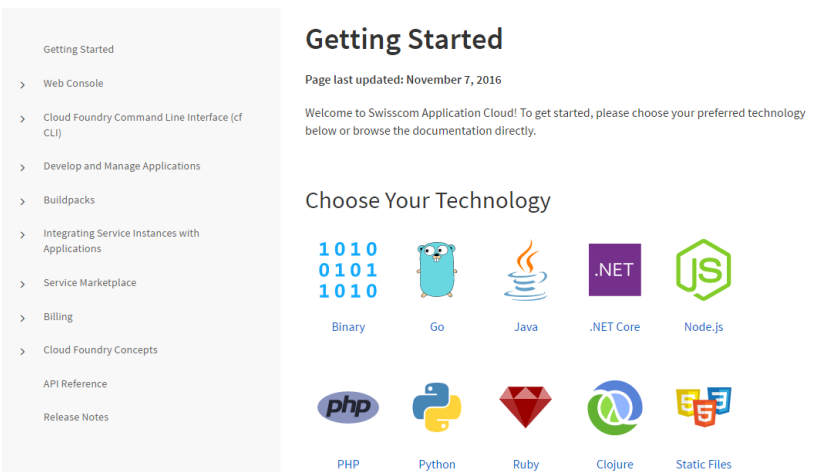
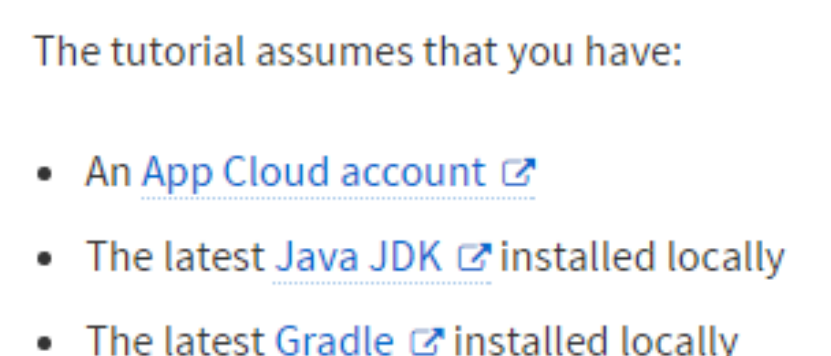
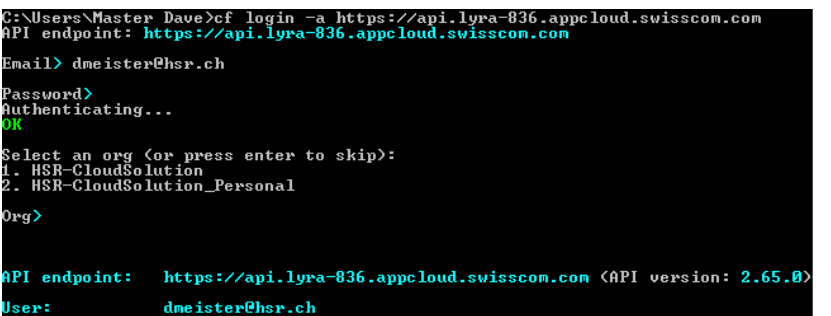
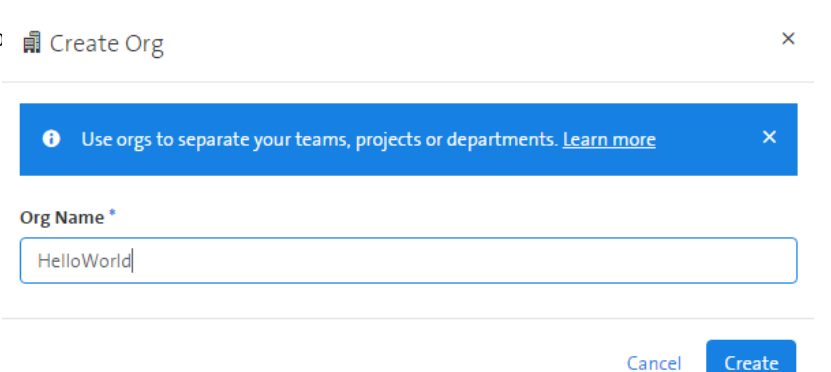
Commands offered by installed plugins:

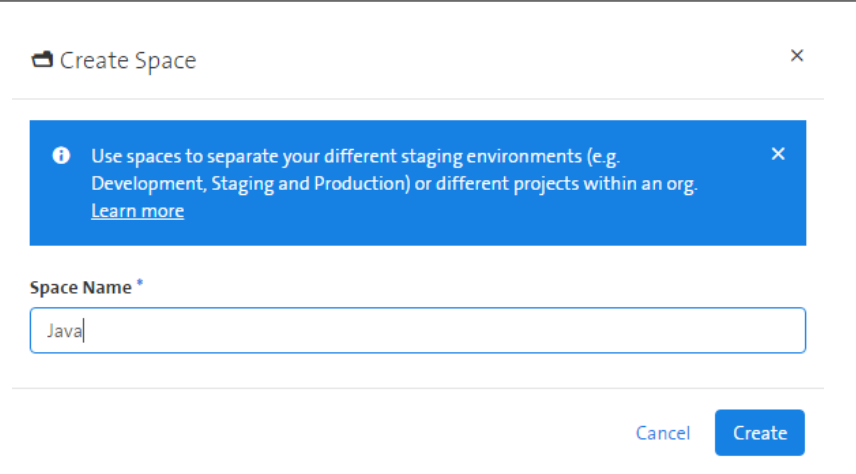
Global options:
  --help, -h Show help
  -v Print API request diagnostics to stdout

These are commonly used commands. Use 'cf help -a' to see all, with descriptions
See 'cf help <command>' to read about a specific command.
```

Nun können wir uns mit der Web Console von der Swisscom App Cloud vertraut machen. Auch hier gibt es ein Tutorial



<p>Jetzt können wir das Tutorial für eine Java App durcharbeiten</p>	
<p>Als Erstes sollte man die Prereqs überprüfen</p>	
<p>Nun muss man sich über die cf Console an der Swisscom App Cloud anmelden. Wir skippen die Zuweisung an eine Org.</p>	
<p>Unter https://console.developer.swisscom.ch erstellen wir eine neue Org Namens „HelloWorld“.</p>	

<p>In dieser Org müssen wir einen Space erstellen.</p>	 <p>The image shows a 'Create Space' dialog box. At the top, it says 'Create Space' with a close button. Below is a blue information banner: 'Use spaces to separate your different staging environments (e.g. Development, Staging and Production) or different projects within an org. Learn more'. Below the banner is a text input field labeled 'Space Name *' containing the text 'Java'. At the bottom right are 'Cancel' and 'Create' buttons.</p>
<p>Wir weisen nun Org und Space mittels „cf target -o HelloWorld -s Java“ zu</p>	<pre>C:\git\Cloud-Solutions_Cloud-Fundamentals\src\cf-sample-app-java>cf target -o HelloWorld -s Java API endpoint: https://api.lyra-836.appcloud.swisscom.com API version: 2.65.0 User: dneister@hsr.ch Org: HelloWorld Space: Java</pre>
<p>Wir kopieren nun die Java Beispielapp von Swisscom</p>	<pre>\$ git clone https://github.com/swisscom/cf-sample-app-java.git Cloning into 'cf-sample-app-java'... remote: Counting objects: 202, done. remote: Total 202 (delta 0), reused 0 (delta 0), pack-reuse 0 Receiving objects: 100% (202/202), 17.69 KiB 0 bytes/s, done. Resolving deltas: 100% (44/44), done. Checking connectivity... done.</pre>
<p>Nun können wir mit Gradle die App bauen. Es wurde ein jar generiert.</p>	<pre>C:\git\Cloud-Solutions_Cloud-Fundamentals\src\cf-sample-app-java>gradle build :compileJava :processResources UP-TO-DATE :classes :jar :assemble :compileTestJava UP-TO-DATE :processTestResources UP-TO-DATE :testClasses UP-TO-DATE :test UP-TO-DATE :check UP-TO-DATE :build BUILD SUCCESSFUL Total time: 10.665 secs</pre>
<p>Diese .jar File können wir nun mittels „cf push my-java-app“ -p <path to jar>, -n „pushen</p>	<pre>C:\git\Cloud-Solutions_Cloud-Fundamentals\src\cf-sample-app-java>cf push my-java-app -p build/libs/cf-sample-app-java-1.0.0.jar -n HelloWorld Updating app my-java-app in org HelloWorld / space Java as dneister@hsr.ch... OK Creating route HelloWorld.scapp.io... OK Binding HelloWorld.scapp.io to my-java-app... OK Uploading my-java-app... Uploading app files from: C:\Users\MASTER\AppData\Local\Temp\unzipped-app186466975 Uploading 3.4M, 2106 files Done uploading OK Starting app my-java-app in org HelloWorld / space Java as dneister@hsr.ch...</pre>

<p>Sobald die App erfolgreich gestartet wurde erscheinen folgende Meldungen</p>	<pre>1 of 1 instances running App started OK App my-java-app was started using this command 'CALCULATED_MEMORY=\$(cat /dev/urandom fold -n 10000 tr -dc 'a-z0-9' fold -w 64 xargs echo sha256sum cut -d ' ' -f 1) java -Xmx100M -Xms64M -Xstack:228k -XmemoryWeights=heap:65,metaspace:10,native:15 -Xstack10 -XmemoryInitials=heap:100%,metaspace:100% -Xstack10 -XtotalMemory=\$MEMORY_LIMIT && JAVA_OPTS="-Djava.io.tmpdir=\$TMPDIR -XX:OnOutOfMemoryError=\$PWD/.java-buildpack/open_jdk_jre/bin/killjava.sh \$CALCULATED_MEMORY" && eval exec \$PWD/.java-buildpack/open_jdk_jre/bin/java \$JAVA_OPTS -cp \$PWD/.com.swisscom.cloud.cloudfoundry.sampleapp.java.ProductService' Showing health and status for app my-java-app in org HelloWorld / space Java as dmeister@hsr.ch... OK requested state: started instances: 1/1 usage: 1G x 1 instances urls: HelloHost.scapp.io last uploaded: Mon Mar 6 16:09:00 UTC 2017 stack: cflinuxfs2 buildpack: java-buildpack=v3.10-https://github.com/cloudfoundry/java-buildpack.git#193d6b7 java-main open-jdk-like-jre=1.8.0_111 open-jdk-like-memory-calculator=2.0.2_RELEASE s state since cpu memory disk detail #0 running 2017-03-06 05:09:50 PM 0.0% 35.8M of 1G 134M of 1G</pre>
<p>Mittels „cf app my-java-app“ kann man den Status der Java App abrufen</p>	<pre>C:\git\Cloud-Solutions_Cloud-Fundamentals\src\cf-sample-app-java>cf app my-java-app Showing health and status for app my-java-app in org HelloWorld / space Java as dmeister@hsr.ch... OK requested state: started instances: 1/1 usage: 1G x 1 instances urls: HelloHost.scapp.io last uploaded: Mon Mar 6 16:09:00 UTC 2017 stack: cflinuxfs2 buildpack: java-buildpack=v3.10-https://github.com/cloudfoundry/java-buildpack.git#193d6b7 java-main open-jdk-like-jre=1.8.0_111 open-jdk-like-memory-calculator=2.0.2_RELEASE s state since cpu memory disk detail #0 running 2017-03-06 05:09:50 PM 0.1% 56.2M of 1G 134M of 1G</pre>
<p>Mittels „cf scale my-java-app“ erhält man eine Übersicht über die verwendeten Ressourcen Memory, Disk, Instances</p>	<pre>C:\git\Cloud-Solutions_Cloud-Fundamentals\src\cf-sample-app-java>cf scale my-java-app Showing current scale of app my-java-app in org HelloWorld / space Java as dmeister@hsr.ch... OK memory: 1G disk: 1G instances: 1</pre>
<p>Horizontales Scaling geschieht mittels „cf scale my-java-app -i 3“. So werden nun 3 Instanzen verwendet</p>	<pre>C:\git\Cloud-Solutions_Cloud-Fundamentals\src\cf-sample-app-java>cf scale my-java-app -i 3 Scaling app my-java-app in org HelloWorld / space Java as dmeister@hsr.ch... OK</pre>

Vertikales Scaling geschieht mittels „cf scale my-java-app -m 2G“. So werden 2 GB RAM verwendet

```
C:\git\Cloud-Solutions_Cloud-Fundamentals\src\cf-sample-app-java>cf scale my-java-app -m 2G
This will cause the app to restart. Are you sure you want to scale my-java-app?
yes
Scaling app my-java-app in org HelloWorld / space Java as dmeister@hsr.ch...
OK
Stopping app my-java-app in org HelloWorld / space Java as dmeister@hsr.ch...
OK
Starting app my-java-app in org HelloWorld / space Java as dmeister@hsr.ch...
0 of 3 instances running, 3 starting
3 of 3 instances running
App started
OK
App my-java-app was started using this command `CALCULATED_MEMORY=${PWD}/.java-buildpack/open_jdk_jre/bin/java-buildpack-memory-calculator-2.0.2_RELEASE -memorySizes=metaspace:64m...stack:228k..-memoryWeights=heap:65,metaspace:10,native:15,stack:10 -memoryInitials=heap:100%,metaspace:100% -stackThreads=300 -totMemory=SMEMORY_LIMIT> && JAVA_OPTS="-Djava.io.tmpdir=$TMPDIR -XX:OnOutOfMemoryError=$PWD/.java-buildpack/open_jdk_jre/bin/killjava.sh $CALCULATED_MEMORY" && eval exec $PWD/.java-buildpack/open_jdk_jre/bin/java $JAVA_OPTS -cp $PWD/.com.swisscom.cloud.cloudfoundry.sampleapp.java.ProductService`
Showing health and status for app my-java-app in org HelloWorld / space Java as dmeister@hsr.ch...
OK
requested state: started
instances: 3/3
usage: 2G x 3 instances
urls: HelloHost.scapp.io
last uploaded: Mon Mar 6 16:09:00 UTC 2017
stack: cflinuxfs2
buildpack: java-buildpack=v3.10-https://github.com/cloudfoundry/java-buildpack.git#193d6b7 java-main open-jdk-like-jre=1.8.0_111 open-jdk-like-memory-calculator=2.0.2_RELEASE
```

#	state	since	cpu	memory	disk	detail
#0	running	2017-03-06 05:20:47 PM	0.1%	72.1M of 2G	134M of 1G	
#1	running	2017-03-06 05:20:47 PM	0.0%	48.1M of 2G	134M of 1G	
#2	running	2017-03-06 05:20:47 PM	0.1%	31.9M of 2G	134M of 1G	

Es gilt nun zu beachten, dass nach dem Scaling die Betriebskosten gestiegen sind.

HelloWorld / java / my-java-app Estimated Monthly Cost: CHF 144.05

my-java-app Started Instances 3 Memory Limit 2048 MB Disk Limit 1 GB

Restart Restage Stop Delete

#	STATUS	SINCE	CPU	MEMORY	DISK
0	Running	2 minutes	0.1%	60 MB	134 MB
1	Running	2 minutes	0.2%	61.9 MB	134 MB
2	Running	2 minutes	0.2%	32.7 MB	134 MB

Die Logfiles können einfach mittels „cf logs my-java-app -recent“ abgerufen werden

```
C:\git\Cloud-Solutions_Cloud-Fundamentals\src\cf-sample-app-java>cf stop my-java-app
Stopping app my-java-app in org HelloWorld / space Java as dmeister@hsr.ch...
OK
```


Mittels „cf stop my-java-app“ kann die App gestoppt werden. Falls sie wieder benötigt wird kann sie mittels „cf start my-java-app“ wieder hochgefahren werden.

```
C:\git\Cloud-Solutions_Cloud-Fundamentals\src\cf-sample-app-java>cf start my-jav
a-app
Starting app my-java-app in org HelloWorld / space Java as dmeister@hsr.ch...
0 of 1 instances running, 1 starting
1 of 1 instances running
App started

OK

App my-java-app was started using this command 'CALCULATED_MEMORY=${PWD}/.java-b
uildpack/open_jdk_jre/bin/java-buildpack-memory-calculator-2.0.2_RELEASE -memory
Sizes=metaspace:64m...stack:228k...-memoryWeights=heap:65,metaspace:10,native:15
...stack:10 -memoryInitials=heap:100%,metaspace:100% -stackThreads=300 -totMemory=
$MEMORY_LIMIT> && JAVA_OPTS="-Djava.io.tmpdir=$TMPDIR -XX:OnOutOfMemoryError=$PW
D/.java-buildpack/open_jdk_jre/bin/killjava.sh $CALCULATED_MEMORY" && eval exec
$PWD/.java-buildpack/open_jdk_jre/bin/java $JAVA_OPTS -cp $PWD/. com.swisscom.c
loud.cloudfoundry.sampleapp.java.ProductService'

Showing health and status for app my-java-app in org HelloWorld / space Java as
dmeister@hsr.ch...
OK

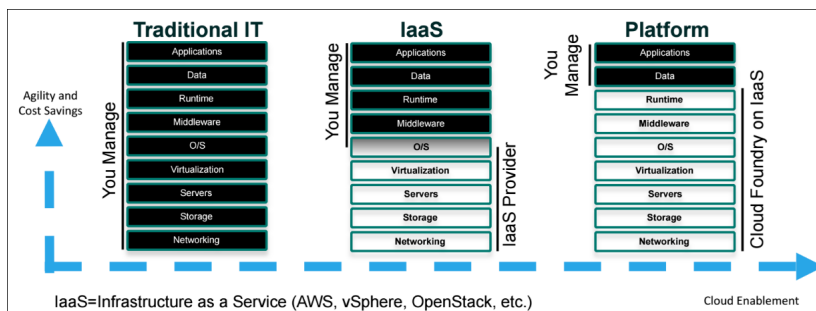
requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: HelloHost.scapp.io
last uploaded: Mon Mar 6 16:09:00 UTC 2017
stack: cflinuxfs2
buildpack: java-buildpack=v3.10-https://github.com/cloudfoundry/java-buildpack.g
it#193d6b7 java-main open-jdk-like-jre=1.8.0_111 open-jdk-like-memory-calculator
=2.0.2_RELEASE

state      since                cpu    memory    disk      details
#0 running  2017-03-06 05:29:24 PM  0.0%  896K of 1G  1.3M of 1G
```

2 OSSM-Definition

3 Cloud Computing Patterns

Die Swisscom Application Cloud setzt das Service-Modell „Platform as a Service“ (PaaS) um. Im Bild sind die drei vorgängigen Servicemodelle im Vergleich mit PaaS ersichtlich.

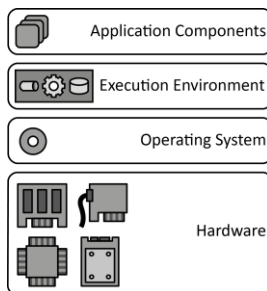


Swisscom verwendet Cloud Foundry für ihre Application Cloud Plattform. Cloud Foundry wird ebenfalls von vielen anderen Providern und Firmen (z.B. IBM, Pivotal, SAP, usw.) verwendet sowie weiterentwickelt. Somit ist Cloud Foundry ein Open Source Projekt. Diese Konstellation verhindert Vendor-lock-ins, d.h. es sollte sehr einfach sein die Provider für die eigenen Enterprise Applikationen auszutauschen. Dies generiert für die Kunden einen echten Mehrwert.

3.1 Process Offering

Als Entwickler möchte man eine fix-fertige Umgebung aus einem Guss. Die Swisscom Application Cloud bietet deshalb fertige Umgebungen für Programmierer.

Diese Anforderungen können mit dem Pattern „Execution Environment“ umgesetzt werden.



Für den Kunden ist vordergründig nicht viel sichtbar. Er kann seine Sources in die Cloud laden und von der Execution Environment ausführen lassen. Die App wird dann in einem Container ausgeführt und die darunterliegende Infrastruktur ist für den Kunden unbekannt.

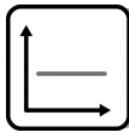
3.2 Workload

Bei der Swisscom Application Cloud wird kein Auto-Scaling unterstützt. Dies verhindert den Einsatz bei gewissen Workload Patterns. Man kann problemlos die Applikation skalieren, jedoch muss dies manuell

geschehen. Eine häufige Änderung des Workloads ist daher ungeeignet. Deshalb fallen folgende Workload Patterns wohl nicht in das Aufgabengebiet der Swisscom Application Cloud:

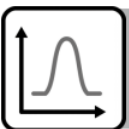
- Periodic Workload
- Unpredictable Workload
- Continuously Changing Workload

Hingegen sind die Workload Patterns „Static Workload“ und „Once-in-a-Lifetime“ Workload gut geeignet.



Static Workload

IT resources with an **equal utilization over time** experience static workload.



Once-in-a-Lifetime Workload

IT resources with an equal utilization over time disturbed by a **strong peak occurring only once** experience once-in-a-lifetime workload.

Bei einer statischen Workload muss offensichtlich nicht skaliert werden. Bei Events, welche einmalig auftreten ist es ebenfalls kein Problem, dies manuell zu skalieren.

3.3 Komponenten

Das Angebot in der Swisscom Application Cloud kann in zwei Gruppen unterteilt werden:

- Applications
- Services

Bei den Applications kann man als User seine Applikationen bereitstellen. Folgende Technologien stehen zur Verfügung:



Binary



Go



Java



.NET Core



Node.js



PHP



Python



Ruby



Clojure



Static Files



Docker



Other

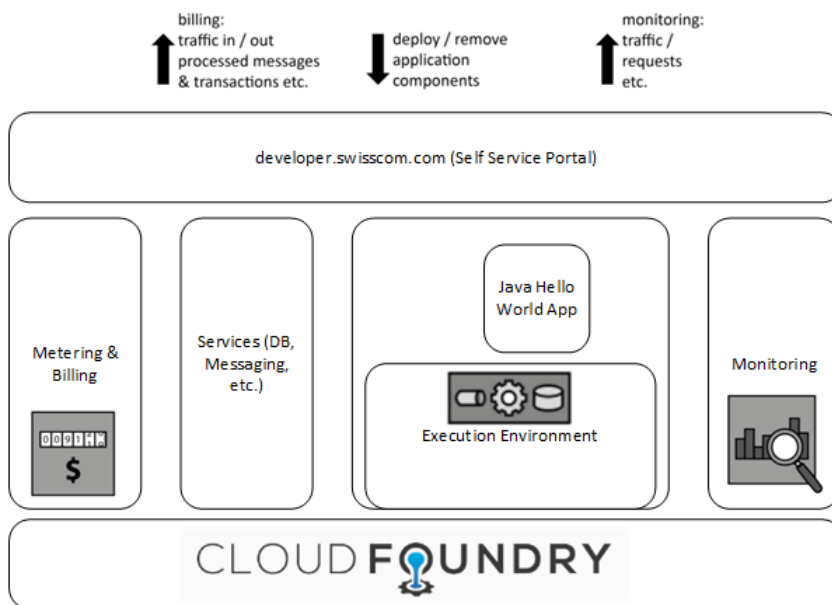
Als Services stehen bekannte Produkte unterschiedlicher Hersteller bereit. Diese wären beispielsweise MongoDB als Vertreter der NoSQL Familie. MariaDB fungiert als SQL Datenbank und Redis als Key-Value Store. Für Queueing und Messaging steht RabbitMQ zur Verfügung.

Swisscom Application Cloud stellt das CCP „Execution Environment“ dem Kunden zur Verfügung. Hypervisor als Process Offering ist an dieser Stelle fehl am Platz da man auf einer höheren Abstraktionsstufe arbeitet. Die Hardware resp. Virtualisierungsplattform ist hier nicht ersichtlich, resp. Black-Box.

Möchte man Map-Reduce Computing, so muss man sich selber darum kümmern, resp. es wird von Swisscom nichts out-of-the-box angeboten.

3.4 Übersichtsgrafik

Die Hello World Java App kann gemäss dem Pattern „Elastic Platform“ platziert werden.



4 Self-Information

5 Preisrecherche

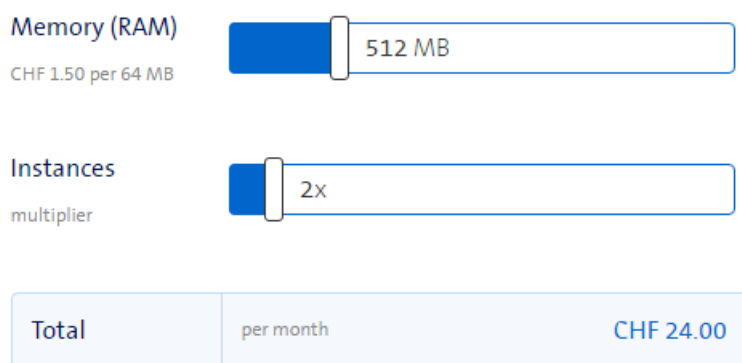
Swisscom verfolgt eine usage-based Pricing Strategie. Man bezahlt also nur für laufende Dienst Die Kosten setzen sich zusammen aus den Ressourcen für die App + zusätzliche Services. Seit Ende 2016 wird bei der Swisscom stündlich abgerechnet. Davor wurde in täglichen Intervallen abgerechnet. Die Preise sind einfachheitshalber aber meist in CHF pro Monat angegeben.

Für die Datenbankprodukte (MariaDB, MongoDB) hat Swisscom die Instanzen in drei verschiedenen Grössen kreiert (S, M, L). Man kann pro Grösse bis zu 10 Instanzen verwenden.

Bei RabbitMQ bezahlt man Messages pro Monat. Für 1 Mio. Messages bezahlt man CHF 13.20.-.

Für Diskspace bezahlt man einen Basispreis von CHF 9.00.-, damit hat man 100GB zur Verfügung. Für jede weiteren 100GB muss man zusätzliche CHF 6.00.- bezahlen.

Das Pricing für die Apps ist sehr einfach gehalten. Man hat nur 2 Faktoren, welche den Preis beeinflussen, nämlich Instanzen und Memory. Man bezahlt pro 64MB CHF 1.50.- pro Monat. Die Instanzen dienen als Faktor. Wählen wir beispielsweise für unsere Applikation 512MB mit 2 Instanzen, so kostet uns das $(512MB/64MB) * 1.50CHF * 2 = 24CHF$ monatlich.



Vor der Erstellung einer App muss dem Account eine Kreditkarte hinterlegt werden. Eine Bezahlung auf Rechnung ist in der Public Application Cloud nicht möglich.

Für neue User steht eine „Welcome Offer“ zur Verfügung. Für die erste 3 Monate werden Kosten bis CHF 100.00.- pro Monat erlassen. So kann man als Interessent ausgiebig testen.

Die Preispolitik ist bei der Swisscom Application Cloud viel einfacher als bei AWS EC2. Die geforderte Konfiguration lässt sich weder mit Swisscom, noch mit AWS so einfach berechnen. Wir gehen nun von einer anderen Konfiguration aus:

- 2 Instanzen
- 512MB RAM pro Instanz
- 20GB Diskspace

Bei Swisscom kostet eine solche Konfiguration CHF 33.00.- monatlich. Bei AWS EC2 USD 11.44. Selbst wenn man die Währungsvorteile vom amerikanischen Dollar aussen vor lässt, ist Amazon doch sehr deutlich günstiger. Der Vergleich ist aber sehr gewagt, da bei Swisscom nicht ersichtlich ist, welche Computing Leistung man erhält. Bei AWS EC2 erhält man für diesen Preis eine sehr schwache CPU Leistung.

6 Preisvergleich Hosting vs IaaS vs PaaS

7 Twelve-Factor Apps

Abbildungsverzeichnis