

## 1 IaaS vs. PaaS

Wir sind beim Vergleich von PaaS zu IaaS zu dem Schluss gekommen, dass Vorteile immer stark in Abhängigkeit des Anwenders und des Anwendungskontextes zu sehen sind. Möchte der Anwender den Fokus auf die Entwicklung und Bereitstellung, so kann das Verwenden von PaaS die Entwicklungszeit und auch Kosten reduzieren, er muss sich nicht um die Wartung / Koordinierung und Erweiterung der Infrastruktur (lies: virtuellen Maschinen) kümmern. Er ist aber gleichzeitig an das Angebot und den Service des Anbieters gebunden.

Vorteil	Nachteil
<ul style="list-style-type: none"><li>• Entwickler kann sich auf Applikation konzentrieren.</li><li>• Keine/Wenig Kenntnisse zur Instandhaltung der Infrastruktur notwendig.</li><li>• Anwender muss sich nicht um CPU/Speicher kümmern. Stellt gewünschte Grenzwerte ein. Management auf IS-Ebene vom PaaS-Provider.</li><li>• Skalierung und Load-Balancing wird vereinfacht. Nur Applikationsschicht muss angepasst werden.</li><li>• Schnelles deployen einer (fertigen) Anwendung. Leichte Verknüpfung mit Release Management (Bluemix und JazzHub).</li></ul>	<ul style="list-style-type: none"><li>• Gebunden an das Angebot des PaaS-Anbieter. Plattformen und Protokolle werden vom Anbieter bestimmt. Erschwert die Migration zwischen verschiedenen Anbietern.</li><li>• Kein Einfluss auf das OS.</li><li>• Kein Einfluss auf aktuellste/gepatchte Versionen (Update-Politik ist von Anbieter bestimmt).</li><li>• Keinen Einfluss auf die Infrastrukturebene. Leistung der Server/VM werden vom Anbieter bestimmt.</li><li>• Keinen Einfluss ob Shared oder Dedicated Maschinen. Man muss Anbieter bei Management vertrauen.</li><li>• Es werden nur Service-Garantien gegeben (CPU, Speicher etc).</li><li>• Bei Verwendung einer Skalierung/LB muss man sich an die Protokolle und Prozesse des Anbieters richten</li></ul>

## 2 Chat und Chat-Log mittels Bluemix

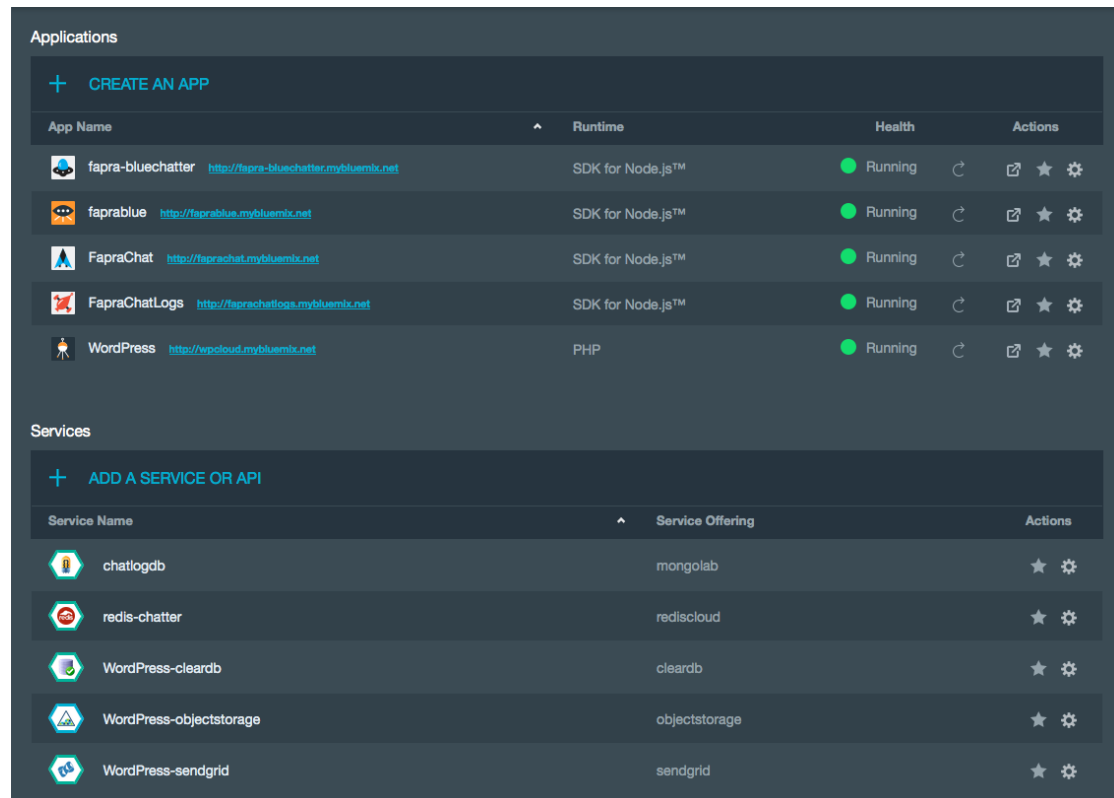
Um den Chat und den Chat-Log mittels Bluemix zu deployen muss zunächst das Cloud-Foundry-Tool `cf` lokal heruntergeladen werden und dem Pfad hinzugefügt werden um das Skript nutzen zu können. Mit diesem Tool kann dann der Chat und der Chat-Log gemäß der Anleitung im GitHub-Repository installiert werden. In unserem Github-Repository befindet sich dafür ein Skript (`install-chat`). Im Header dieses Skripts können die Konfigurationen für Bluemix und die zu installierenden Apps geändert werden. Das Skript gibt am Ende die Hosts für den Chat und den Chat-Log aus.

Der Chat-Log muss nach dem Senden der ersten Chat-Nachricht mittels des Befehls `cf restart <LOG-APP-NAME>` neugestartet werden um die Chats einsehen zu können. Die Chatdatenbank muss anscheinend durch das Senden der ersten Nachricht erzeugt oder installiert werden.

Beim Skalieren der Chat-App muss beachtet werden, dass die einzelnen Instanzen nicht miteinander verbunden sind, sodass man bei mehr als einer Instanz zufällig mit einer verbunden wird. Das bedeutet insbesondere, dass man nicht zwangsläufig mit der Instanz verbunden wird, mit der der Chat-Partner verbunden wurde. Dennoch werden alle Chats in derselben Datenbank gespeichert. In der Chat-Datenbank wird nicht gespeichert, aus welcher Instanz die Chat-Nachrichten stammen. Es ist so also nur mit Informationen aus der Datenbank nicht möglich, einzelne Gesprächsverläufe zu rekonstruieren.

Das Deployment der App ist nur in der US und nicht in der EU-Region möglich, da der Datenbankserver nur in der US-Region zur Verfügung steht.

Unsere deployten Anwendungen sind unter <http://faprachats.mybluemix.net/> und <http://faprachatslogs.mybluemix.net/> zu finden.



The screenshot shows the Bluemix console interface. At the top, there's a section for 'Applications' with a '+ CREATE AN APP' button. Below it is a table of running applications. At the bottom, there's a section for 'Services' with a '+ ADD A SERVICE OR API' button. Below it is a table of available services.

App Name	Runtime	Health	Actions
fapra-bluechatter <a href="http://fapra-bluechatter.mybluemix.net">http://fapra-bluechatter.mybluemix.net</a>	SDK for Node.js™	Running	Refresh, Link, Star, Settings
faprablue <a href="http://faprablue.mybluemix.net">http://faprablue.mybluemix.net</a>	SDK for Node.js™	Running	Refresh, Link, Star, Settings
FapraChat <a href="http://faprachat.mybluemix.net">http://faprachat.mybluemix.net</a>	SDK for Node.js™	Running	Refresh, Link, Star, Settings
FapraChatLogs <a href="http://faprachatlogs.mybluemix.net">http://faprachatlogs.mybluemix.net</a>	SDK for Node.js™	Running	Refresh, Link, Star, Settings
WordPress <a href="http://wpcloud.mybluemix.net">http://wpcloud.mybluemix.net</a>	PHP	Running	Refresh, Link, Star, Settings

Service Name	Service Offering	Actions
chatlogdb	mongolab	Star, Settings
redis-chatter	rediscloud	Star, Settings
WordPress-clearadb	clearadb	Star, Settings
WordPress-objectstorage	objectstorage	Star, Settings
WordPress-sendgrid	sendgrid	Star, Settings

Abbildung 1: Laufende Anwendungen und Dienste in Bluemix

### 3 Weitere Anwendung mit Bluemix

Da die Aufgabe explizit WordPress nennt, haben wir uns zuerst dies angesehen. Es stellte sich heraus, dass WordPress einst nicht ganz simpel auf Bluemix zu nutzen war, was zum einen an einer Art fehlendem, fertigen LAMP Stack lag, zum anderen an einer nicht existenten, persistenten Dateispeichermöglichkeit in Bluemix.

Dies hat sich jedoch geändert und WordPress lässt sich seit letztem Dezember nun enorm simpel in Bluemix aufsetzen. Es sind essentiell nur noch wenige Klicks in der grafischen Oberfläche notwendig, was wir so auch erfolgreich durchführten. Realisiert wird dies mittels eines sogenannten „Boilerplate“ Pakets, welches praktisch ein Set an Services beinhaltet. Abbildung 1 zeigt die Übersicht der laufenden Anwendungen und Dienste, die untere Anwendung und die unteren drei Dienste werden beim Ausbringen des Boilerplate Pakets erzeugt. Es werden darüber hinaus Plugins angeboten, die eine Verknüpfung von Wordpress mit beispielsweise IBM Object Storage ermöglicht, um Metadaten persistent zu speichern. Unsere WordPress-Instanz ist unter <http://wpcloud.mybluemix.net/> zu erreichen.

Anschließend widmeten wir uns den beiden genannten Beispielanwendungen. Da „todo-apps“ auf den ersten Blick eine ähnliche Komplexität wie WordPress aufzuweisen scheint

(es kommt mit fertigen Deployment-Skripten), haben wir folgend „bluechatter“ deployed.

Dazu wurde das Script `deploy-bluechatter.sh` genutzt. Dieses startet nach dem Login einen Redis Datenbankservice, klonet das bluechatter Respository und pusht dieses, nach einer kleinen Modifikation des Redis "Plans" (der im Originalrepo genutzte existiert nicht mehr), zu Bluemix.

Weitere Befehle, wie in 4.3, sind nicht notwendig, da die Relationen und Konfigurationen des/der Service(s) und Application(s) in der `manifest.yml` bereits definiert sind. Dieser Ansatz der Konfiguration empfiehlt sich im allgemeinen für EëchtweltProjekte, speziell wenn die Konfiguration (ver|ge)teilt werden können soll. Der so deployte Chat kann unter <http://fapra-bluechatter.mybluemix.net/> erreicht werden.

## 4 Bluemix und JazzHub

Um Bluemix und JazzHub auszuprobieren haben wir wieder die Beispielanwendung *bluechatter* aus Aufgabe 4 genutzt. Nach dem Einloggen mit der IBM-Identität von Bluemix kann ein GitHub-Konto mit JazzHub verknüpft werden. Ein existierendes Repository von GitHub kann dann als App-Repository zu JazzHub hinzugefügt werden. Im Bereich „Build and Deploy“ kann dann im Projekt eine sogenannte Phase erstellt werden. Diese teilt sich in einen Build- und Deploy-Schritt. Gleichzeitig kann in dieser eingestellt werden, dass diese Phase jedes Mal ausgeführt wird, sobald in einen bestimmten Branch des Repositorys eine Änderung durchgeführt wird. Die so deployte Anwendung kann unter <http://faprablue.mybluemix.net/> aufgerufen werden. Das zugehörige GitHub-Repository ist <https://github.com/CloudFapra/bluechatter>.