Einer der Plattformdienste

Micro-Cloud mit Cloud Foundry

Oliver Wolf, Phillip Ghadir

In dieser Ausgabe stellen wir mit Cloud Foundry einen Plattformdienst vor, mit dem Java-Anwendungen in der Cloud betrieben werden können. Die Plattform stellt Java-Anwendungen eine virtuelle Umgebung mit Ressourcen zur Verfügung. Wir haben Cloud Foundry für diese Kolumne gewählt, weil es die Cloud-Plattform auf den eigenen Rechner holt. Dadurch wird die Cloud-Entwicklung netzunabhängig.

Bereits in [Gha12] haben wir mit Red Hat OpenShift Express einen Plattformdienst für Java-Anwendungen vorgestellt. Nun, ein Jahr später, haben wir mit Cloud Foundry ein anderes Angebot herausgepickt und stellen vor, warum es sich lohnt, auch dieses anzuschauen.

Cloud Foundry unterstützt die auf der JVM aufsetzenden Frameworks Spring, Scala, Play und Grails sowie auch Node.js und Ruby, Rails- und Sinatra-Applikationen. Die Applikationen können innerhalb der Cloud Foundry auf Services wie MongoDB, MySQL, RabbitMQ, Redis und vFabric Postgres zugreifen, die von der Plattform bei entsprechender Konfiguration automatisch bereitgestellt werden.

Die Cloud auf dem eigenen Rechner

Cloud Foundry bietet neben der Cloud im Internet eine sogenannte Micro-Cloud an, die in einer VMware VM auf dem eigenen Rechner betrieben wird. Für Windows und Linux-Anwender genügt ein kostenfrei verfügbarer VMware Player. Mac-Anwender hingegen müssen für den dauerhaften Einsatz eine kostenpflichtige Lizenz für VMware Fusion erwerben.

Wer also einen leistungsfähigen Entwicklungsrechner besitzt, kann auch ohne ständigen Netzzugang auf der Cloud-Foundry-Plattform entwickeln. Nachdem die circa 1,5 GB umfassende Micro-Cloud-VM heruntergeladen ist, kann's losgehen.

Entwicklungsumgebung

Freunde grafischer IDEs können die Cloud Foundry Integration Extension mit der Spring Tool Suite oder direkt mit Eclipse verwenden. Darüber hinaus unterstützt die aktuelle Version von Intellij IDEA 12 in der Ultimate-Edition ebenfalls Cloud Foundry. Die Plattform lässt sich aber auch über die Kommandozeile mit dem Werkzeug vmc steuern. Dies ist der in diesem Artikel verwendete Weg.

Sind in der lokalen Entwicklungsumgebung die Werkzeuge für den Einsatz von Cloud Foundry installiert, kann die lokale Umgebung mittels

```
vmc target <cloud-instance-url>
```

mit einer Cloud-Instanz verbunden werden, in die dann im Folgenden die Software deployt werden soll.

Die Entwicklung erfolgt in der gewohnten lokalen Umgebung. Der Entwicklungszyklus muss sich dabei in keiner Weise von der sonstigen Entwicklung unterscheiden. Nach dem Bauen der Software schiebt man sie mit dem Werkzeug vmc einfach in die Cloud. Je nach lokaler Konfiguration kann das lokal oder remote erfolgen.

Der Kasten „Aufsetzen der Micro-Cloud in der VM" skizziert grob, wie Cloud Foundry in der Entwicklungsumgebung aufgesetzt wird. Beim Deployment in einer frisch aufgesetzten Umgebung wird deutlich, dass die Cloud mehr bietet, als eine einfache Virtuelle Maschine mit Server-Betriebssystem. Abbildung 1 stellt die ersten Entwicklungsschritte dar. Wird mit

```
vmc push <deployment-unit>
```

ein Softwarestand in die Cloud geschoben, erkennt Cloud Foundry mittels Introspektion, von welchem Technologie-Stack die Software abhängt. Für die erkannten Abhängigkeiten erfragt `vmc push` relevante Konfigurationsparameter und stellt die passende Umgebung bereit, um unsere Software darin zu deployen.

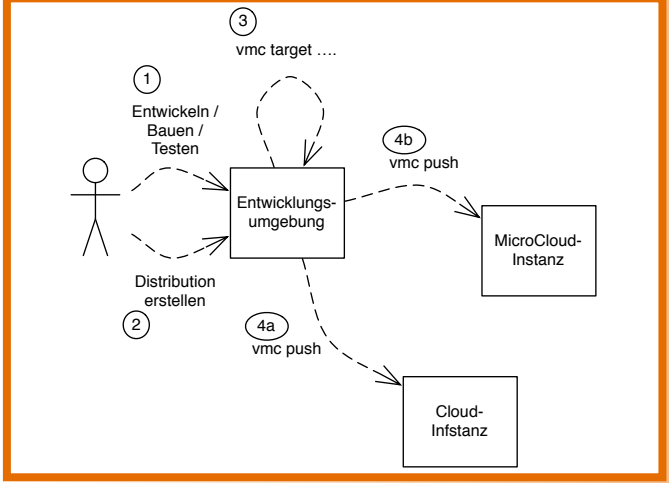

Abb. 1: Entwicklungszyklus mit Cloud Foundry und vmc

Erste Schritte

Wir verwenden für den Einstieg ein mit Play 2 realisiertes Beispiel [CFEx], um die Arbeit mit Cloud Foundry zu demonstrieren. Die kleine Todo-Listen-Anwendung speichert die Todos in MongoDB. Das Technologie-Beispiel ist neben verschiedenen anderen explizit dazu gedacht, die ersten Schritte mit der Cloud zu erleichtern.

Voraussetzung für die folgenden Schritte ist eine installierte Version des Play 2-Frameworks. Zunächst erzeugen wir uns mit

```
git clone https://github.com/cloudfoundry-samples/todolist-play-java-
  mongodb.git
```

eine lokale Kopie des Git-Repositories, in dem der Quellcode der Beispiel-Anwendung liegt. Nun müssen wir die Konfigurationsdatei (app/application.conf) für das Deployment in die Cloud anpassen: Da die Verbindungsparameter der MongoDB in der Cloud erst von der Cloud Foundry vergeben werden, müssen wir Platzhalter eintragen, die beim Deployment automatisch durch die korrekten Werte ersetzt werden. Folgende Platzhalter stehen hier zur Verfügung:

```
mongo.dbname=${?cloud.services.mongodb.connection.db}
mongo.remote.hostname=${?cloud.services.mongodb.connection.hostname}
mongo.remote.port=${?cloud.services.mongodb.connection.port}
mongo.remote.username=${?cloud.services.mongodb.connection.username}
mongo.remote.password=${?cloud.services.mongodb.connection.password}
```

Aufsetzen der Micro-Cloud in der VM

Vor dem Herunterladen der Micro Cloud Foundry (MCF) müssen zunächst ein Cloud-Foundry-Account angelegt und ein eindeutiger Domainname gewählt werden, unter dem die MCF später erreichbar sein wird (z. B. derpraktiker.cloudfoundry.me). Im Gegenzug erhalten wir ein Konfigurationstoken, das wir gleich zur Konfiguration der MCF benötigen.

In dem heruntergeladenen ZIP-Archiv befindet sich eine vmx-Datei, die im VMware-Player (oder in VMWare Fusion für Mac-User) gestartet wird. Das System fährt hoch und zeigt nach dem Booten ein Menü (s. Abb. 2).

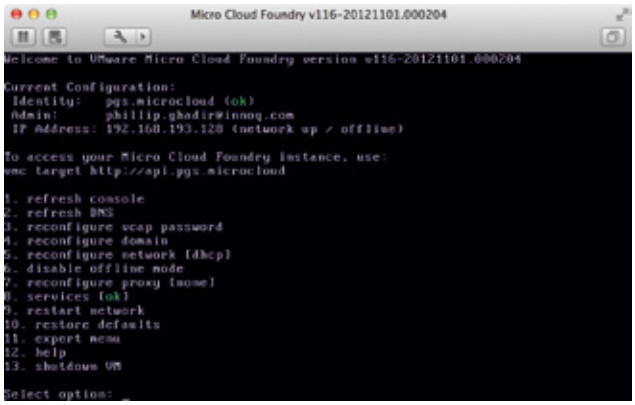

Abb. 2: Boot-Menü

Beim ersten Start müssen wir die MCF zunächst mit der Option `1` konfigurieren: Wir setzen ein Passwort unserer Wahl, wählen die Option `DHCP`, tragen die Adresse unseres HTTP-Proxy ein (sofern notwendig) und geben zu guter Letzt noch unser Konfigurationstoken an. Die MCF baut jetzt eine Verbindung zum Cloud-Foundry-Server auf (deswegen der HTTP-Proxy) und konfiguriert sich selbst. Nach einer kurzen Wartezeit erscheint erneut ein Menü und die MCF ist gestartet. Alle zuvor eingegebenen Konfigurationsparameter lassen sich übrigens jederzeit über die verschiedenen Menüoptionen auch wieder ändern.

Um Programme in die MCF deployen zu können, müssen wir jetzt noch einen Benutzeraccount anlegen. Dazu brauchen wir das Kommandozeilenwerkzeug „vmc“, dessen Installation in der Online-Dokumentation beschrieben ist [CFD]. Zunächst gilt es, die MCF als Zielplattform zu definieren. Dazu dient das Kommando `vmc target [url]`, also beispielsweise `vmc target api.derpraktiker.cloudfoundry.me`. Anschließend können wir mit `vmc register` einen Account anlegen, wobei nach der E-Mail-Adresse (als Benutzername) und einem Passwort gefragt wird. Jetzt ist die MCF soweit einsatzbereit.

In der lokalen Umgebung müssen wir noch mehr konfigurieren, um die MCF offline benutzen zu können – zum Beispiel auf einer Zugfahrt. Standardmäßig wird für jede MCF ein globaler DNS-Eintrag erzeugt, unter dem die (lokale) IP-Adresse der MCF auffindbar ist. Vorteil dabei: Jeder Entwickler im lokalen Netz kann die MCF auf unserem Rechner unter der zuvor gewählten Adresse erreichen. Der Nachteil: Das geht natürlich nur mit funktionierender Verbindung zum Internet. Abhilfe schaffen wir, indem wir die MCF mit der Menüoption `6` in den Offline-Modus setzen. Damit die Namensauflösung trotzdem auf unserem lokalen Rechner noch funktioniert, müssen wir noch eine kleine Anpassung an unserer DNS-Konfiguration vornehmen. Die notwendigen Schritte sind je nach Betriebssystem unterschiedlich und in der Online-Dokumentation beschrieben. Alternativ erledigt das folgende Kommando das auch automatisch:

```
vmc micro offline –-vmx [Pfad zur vmx-Datei]
```

Nun können wir unsere Anwendung mit „play dist“ bauen und paketieren. Mit einem anschließenden

```
vmc push --path=dist/todolist-java-mongodb-1.0-SNAPSHOT.zip
```

starten wir den Deployment-Vorgang, bei dem noch eine Reihe von Parametern abgefragt werden. Als Erstes müssen wir einen Namen für die Anwendung wählen, z. B. „todolist“. Dieser Name dient dazu, bei Aktionen mit dem „vmc“-Werkzeug die Anwendung zu identifizieren (z. B. zum Starten, Stoppen, Löschen usw.). Bei der Wahl des Namens verzichtet man am Besten auf die Verwendung von Punkten („.“).

Im nächsten Schritt erkennt vmc korrekt, dass es sich um eine Play-Anwendung handelt, sodass wir die Frage mit „Y“ beantworten können.

Nun können wir die URL bestimmen, unter der die Anwendung erreichbar sein soll. Als Default wird die URL aus dem Namen der Anwendung und dem Domainnamen unserer MCF (Micro Cloud Foundry) gebildet. So lange wir auf die MCF deployen, sind wir in der Namenswahl frei. Später in der „großen“ Cloud kann es durchaus zu Namenskollisionen mit anderen Anwendern kommen, sodass wir hier den Default unter Umständen ändern müssen.

Anschließend wird die Speicherkonfiguration abgefragt. Der vorgegebene Wert wird automatisch anhand der Art der Anwendung (Programmiersprache, Framework usw.) ermittelt und passt in unserem Fall. Auch die folgende Frage nach der Anzahl der Instanzen beantworten wir mit der vorgegebenen Antwort „1“.

Das Bildschirmfoto in Abbildung 3 zeigt die interaktiven Fragen des vmc-Aufrufs. Insgesamt dauert das Deployment nach dem Beantworten aller Fragen für die kleine Play-Anwendung nur wenige Sekunden auf einem aktuellen Notebook.

Abbildung 4 zeigt das zufrieden stellende Ergebnis des kleinen Exkurses. Nach der Konfiguration der Umgebung, dem Auschecken der Quellen, dem Bauen der Anwendung und dem Hochladen in die Cloud steht die Anwendung unter der beim Deployment angezeigten URL zur Verfügung.

Sollte anstatt der Todo-Listen-Anwendung eine Fehlermeldung oder Exception zu sehen sein, hilft vielleicht ein Blick in die Log-Datei. Mit dem Kommando

```
vmc logs todolist
```

lässt sich diese einsehen. Beendet wird die Anwendung mit

```
vmc stop todolist
```

und mit

```
vmc delete todolist
```

kann sie wieder komplett von der MCF entfernt werden.

Soll das Deployment anstatt in die lokale MCF in die „große“ Cloud Foundry im Internet erfolgen, sind die Schritte identisch. Lediglich die Zielumgebung muss mit

Abb. 3: Der Screenshot zeigt die Fragen beim Deployment in die Cloud

Abb. 4: Screenshot der Anwendung TodoList - live aus der MCF

```
vmc target api.cloudfoundry.com
```

umgeschaltet werden.

Stand der Dokumentation

Derzeit beantwortet die Dokumentation viele naheliegende Fragen noch nicht. In vielen Fällen ist man auf die Suche auf StackOverflow zu Cloud Foundry angewiesen [CFQu].

Zwar wird man teilweise recht schnell fündig, aber manche Wege hätten wir uns lieber erspart. Mit einer etwas ausgereifteren Benutzer-/Entwickler-Dokumentation hätten wir uns gleich von Anfang an an die Konventionen der Plattform gehalten und zum Beispiel Namen ohne „.“ verwendet bzw. die korrekten Schalter verwendet, um die Unterschiede zwischen der Java-Version in der Entwicklungsumgebung (Java 7) und der Cloud-Instanz (Java 6) auszugleichen.

Fazit

Cloud Foundry ist ein Angebot, das für verschiedene Frameworks eine Laufzeitumgebung bereitstellt, die lokal ähnlich wie in der Cloud laufen kann. Aufgrund der Ausrichtung erfordert Cloud Foundry eine Menge an Verständnis über die Abläufe und Zusammenhänge zwischen Build-Infrastruktur, Paketen, Konfigurationen und den Eigenheiten der in einer Cloud-Instanz bereitgestellten Infrastruktur.

Die Bereitstellung der Services erfolgt einfach und gelungen. Der Zugriff auf die Umgebung aus der eigenen Applikation heraus ist definiert, sodass hier eine dynamische Konfiguration unproblematisch bleiben sollte.

An verschiedenen Stellen sollten Konventionen eingehalten werden, die so noch nicht allgemein dokumentiert sind, sondern aus Forenbeiträgen ermittelt werden müssen. Unser subjektiver Eindruck ist, dass Cloud Foundry etwas mehr Anforderungen an die Deployment-Pakete stellt, als wir dies bislang gewohnt waren.

Services in der Cloud Foundry

Daten in der Cloud

Applikationen, die in Cloud Foundry betrieben werden, können auf eine Reihe von Diensten zurück greifen. Zurzeit werden für die Datenhaltung mehrere relationale und NoSQL-Datenbankmanagementsysteme unterstützt. Für die Verwaltung von relationalen Daten stehen in der Plattform MySQL und vFabric Postgres zur Verfügung. Wer eher Schlüssel-/Wertpaare verwaltet, hat mit Redis eine passende Lösung. Für Dokumenten-orientierte Daten bietet sich auf Cloud Foundry MongoDB an. Anwendungen, die Messaging in der Cloud benötigen, steht RabbitMQ zur Verfügung.

Service-Instanziierung in der Cloud

Services können in der Cloud instanziiert werden. Der einfachste Weg ist das automatische Erzeugen der erforderlichen Service-Instanzen während des Deployments einer Anwendung. Wird über `vmc push ...` eine Anwendung in die Cloud geschoben, die über konfigurierte Abhängigkeiten verfügt, kann man auf die Frage *Create services to bind to ‚<app-name>‘* die entsprechenden Services instanziieren und bei Bedarf auch selbst benennen.

Konfiguration der Applikation

Cloud Foundry kann die Konfiguration der Anwendung automatisch anpassen, indem sie beim Deployment Platzhalter durch die konkreten Konfigurationsparameter ersetzt. Diese Auto-Rekonfiguration ist je nach Anwendungsart unterschiedlich ausgereift. Bei der Verwendung von Play 2 mussten wir, wie im Text geschildert, die Konfigurationswerte durch Platzhalter ersetzen.

Möchten wir die Konfiguration programmatisch selbst ermitteln, kann die entsprechende lokale Konfiguration der in der Cloud-Instanz für die Anwendung bereitgestellten Services aus der Umgebungsvariable `VCAP_SERVICES` abgefragt werden. Cloud Foundry legt in der Umgebungsvariable die der Anwendung zugewiesenen Services samt Verbindungsparameter ab. Der Variablen-Inhalt ist ein Service-Dokument im json-Format, das gruppiert nach Service-Typ (MongoDB, MySQL, Redis usw.) die Liste von Zugangsinformationen, Namen und weiteren Parametern enthält.

Naturgemäß erfordert die MCF etwas mehr Grundverständnis und bietet dafür aber sowohl einen Online- als auch einen Offline-Betrieb.

Literatur und Links

[CFD] Cloud-Foundry-Dokumentation inkl. Installationsanleitung und Getting Started,
`http://docs.cloudfoundry.com`
[CFEx] auf Play 2 basierende Todo-List-Beispiel-Anwendung für Cloud Foundry,
`https://github.com/cloudfoundry-samples/todolist-play-java-mongodb`
[CFH] Cloud Foundry Homepage, `http://www.cloudfoundry.com`
[CFQu] Stack Overflow zu Cloud Foundry,
`http://stackoverflow.com/questions/tagged/cloudfoundry`
[Gha12] Ph. Ghadir, OpenShift Express, in: JavaSPEKTRUM, 1/2012

Oliver Wolf ist Principal Consultant bei der innoQ Deutschland GmbH. Neben Softwarearchitektur und agilen Softwareentwicklungsmethoden gilt sein Interesse der engeren Verbindung von Entwicklung und Betrieb.
E-Mail: oliver.wolf@innoq.com

Phillip Ghadir baut am liebsten tragfähige, langlebige Softwaresysteme. Er ist Mitglied der Geschäftsleitung bei innoQ und hat sich früh auf Architekturen für verteilte, unternehmenskritische Systeme spezialisiert. Darüber hinaus ist er Mitbegründer und aktives Mitglied des iSAQB, des International Software Architecture Qualification Board.
E-Mail: phillip.ghadir@innoq.com