Java EE in der Cloud

OpenShift Express

Phillip Ghadir

Oracle hat bereits zu Zeiten des Grid-Computing-Hypes verkündet, dass die bestehenden Java EE/J2EE-Standards völlig ausreichen, um Systeme für das Grid zu entwickeln. Diese Idee gewinnt im Cloud Computing bei verschiedenen „Platform as a Service (PaaS)"-Anbietern an Bedeutung, zumal der vom JCP akzeptierte JSR 342 verschiedene Features für Java EE 7 vorschlägt, die effizientere PaaS ermöglichen. Einen der Anbieter habe ich herausgepickt und mit ihm erste Geh-Versuche gemacht.

Lose Kopplung zum PaaS-Anbieter – auch rechtlich?

Wenn wir unsere Software auf einer Cloud-Plattform ausführen lassen wollen, sollten wir sehr genau darauf achten, welche Verpflichtung wir eingehen und welche Zugeständnisse man von uns verlangt. Zwar beginnen die Angebote typischerweise gebührenfrei und können bei Bedarf flexibel (und dann kostenpflichtig) aufgestockt werden, aber dafür haben wir – nach meinen bescheidenen juristischen Vorurteilen – dennoch einen Vertrag abzuschließen. Wer einen Plattformdienst einsetzen möchte, sollte daher die Vertrags-/Nutzungsbedingungen genau prüfen.

Eigentlich beinhalten alle Cloud-Angebote eine Klausel, die besagt: „Wer sich für den Dienst anmeldet, akzeptiert alle Bedingungen, die nachgelesen werden können." – Und da uns lose Kopplung eigentlich am Herzen liegen sollte, sollten wir auch sicherstellen, dass die Kopplung zur Plattform, die wir auswählen, keine unerwünschten Konsequenzen mit sich bringt.

PaaS-Angebote für Java (EE)

Geva Perry hat in seinem Blog [gevap] kürzlich eine nette Übersicht über Cloud-Plattformen für die Java-Welt zusammengestellt. Die Liste beinhaltet Referenzen auf die Cloud-Angebote verschiedener bereits etablierter Anbieter sowie einiger Start-Ups. Eine Bewertung oder Gegenüberstellung liefert der BlogPost nicht. Die folgende Liste von PaaS-Anbietern ist sicherlich nicht vollständig:

- Amazon Elastic Beanstalk
- CloudBees Run@Cloud
- Cumulogic
- Google App Engine
- Hivext Technologies Jelastic
- IBM SmartCloud Application Services
- Microsoft Azure
- Oracle Public Cloud
- OutSystems Agile Platform
- Red Hat OpenShift
- Salesforce.com Heroku for Java
- VMWare CloudFoundry
- WSO2 Stratos/StratosLive

Interessante PaaS-Anbieter für Java EE

Wer seine Java EE-Software in der Cloud betreiben lassen möchte, kann den Service unter anderem von den im Kasten „PaaS-Angebote für Java (EE)" Aufgeführten beziehen. In diesem Artikel greifen wir einen Service heraus, der auf unterschiedliche Weise in den Entwicklungsprozess eingebunden werden kann, aber ansonsten praktisch keine speziellen Anforderungen an die Softwarearchitektur unseres Systems erfordert: Red Hat OpenShift.

OpenShift – mit Git von der IDE in die Cloud

Red Hat bietet mit *OpenShift Express* einen einfachen Cloud-Dienst an, der sich insbesondere für die Erstellung eines neuen Systems eignet. Konfigurationsmöglichkeiten in Bezug auf Skalierbarkeit sind der größeren Schwester *OpenShift Flex* vorbehalten.

OpenShift Express verwendet einen Workflow, der sich an der Arbeitsweise von Heroku orientiert, einem ursprünglich auf die Ruby on Rails-Community ausgerichteten Cloud-Anbieter. Über einfache Kommandos initialisiert man die Cloud-Umgebung, erstellt neue Projekte und lässt die Plattform automatisch die Software deployen, sobald die Quelltextänderungen ins entfernte Versionsverwaltungssystem eingecheckt wurden.

Versionsverwaltung mit Git

Es gibt im Netz unzählige Tutorien, deshalb möchte ich in diesem Artikel nicht darauf eingehen. Dennoch muss ich ganz kurz zwei der vielen Unterschiede zwischen Git und solchen Werkzeugen wie Subversion darstellen:

Ein Git-Repository ist stets vollständig. Während Subversion eine 1:n-Beziehung zwischen Repository und Arbeitskopie hat, gibt es bei Git eine n:m-Beziehung zwischen Git-Repositories und dann jeweils eine 1:1-Beziehung zwischen Git-Repository und Arbeitskopie. Praktisch jede Änderung an der Arbeitskopie führt dazu, dass man erneut die geänderten Dateien und Verzeichnisse zum Repository hinzufügen muss.

Mittels `git add <Datei>` und `git commit <Datei>` checkt man ins lokale Git-Repository ein. Mittels `git push/merge/pull` synchronisiert man verschiedene Git-Repositories. Und mit `git clone/init` legt man lokale Git-Repositories an.

Da OpenShift zur Synchronisation einen auf Git basierenden Workflow (s. Kasten „Versionsverwaltung mit Git“, [git]) einsetzt, fühlt sich die Arbeit mit OpenShift für Git-erfahrene Entwickler sehr natürlich und flüssig an. Der Roundtrip durch Re-Deployments funktioniert (für kleinere Systeme) zügig. Wer mit OpenShift entwickelt, kann sich schnell an die Synchronisation im Hintergrund gewöhnen.

Alle, die immer noch auf Continuous/Daily/Nightly-Integration verzichten, sehen mit OpenShift sehr einfach und schnell, welchen Charme so eine direkte Commit->Build->Deploy-Schleife hat.

Erste Schritte mit OpenShift Express

Nachdem wir das Kleingedruckte auf der OpenShift-Website gelesen und als unkritisch eingestuft haben, können wir uns bei OpenShift ein Benutzerkonto einrichten. Mit diesem Konto können wir dann sehr einfach Anwendungen anlegen, die wir mit den üblichen Mitteln entwickeln und testen können.

1.) Anmelden

Über die OpenShift-Webseite [rhc] kann man sich ohne Weiteres ein Benutzerkonto anlegen. Ich wiederhole mich: Bitte prüfen Sie, ob Sie die Nutzungsbedingungen und das Kleingedruckte akzeptieren wollen (u. a. können Sie das Benutzerkonto nicht wieder löschen).

Nach der Bestätigung der Anmeldung können die Client-Tools installiert werden. Für die Betriebssysteme Linux, Windows und OSX ist eine Anleitung direkt nach der Anmeldung verfügbar.

2.) Domäne anlegen

OpenShift verwendet ein Domänenkonzept. Dank weltweit eindeutigem Domänennamen können Sie Ihre Applikationen benennen, wie Sie es möchten. Wir können die Domäne entweder über das Kommando-Zeilen-Werkzeug **`rhc-create-domain`**

```
rhc-create-domain -n <Domänen-Name>-l <ihr rhc-login>
```

oder aber über das Control Panel auf der Webseite anlegen

3.) Projekt anlegen

Ein Projekt können wir einfach über die Weboberfläche (über den Link „Control Panel“) oder aber über das Shell-Kommando:

```
rhc-create-app -a<app-name> -t <type>
```

anlegen. **`<app-name>`** ist ein beliebiger alphanummerischer Name und **`<type>`** kann derzeit einer der folgenden Strings sein: raw-0.1, jenkins-1.4, jbossas-7.0, perl-5.10, rack-1.1, wsgi-3.2, php-5.3.

Welche Anwendungstypen zurzeit unterstützt werden, kann entweder auf dem Control Panel auf der Weboberfläche in der Auswahl gesehen werden oder aber über das Shell-Kommando:

```
rhc-create-app --help
```

Legt man mit obigem **`rhc-create-app`**-Aufruf eine Anwendung an, erhält man im Anschluss die Ausgabe auf der Console, unter welcher URL die Anwendung aufgerufen werden kann und wo das Git-Repository zu finden ist.

4.) Projekt einchecken

Mithilfe eines Git-Clients können wir den Projektstand nun vom Repository abrufen und bearbeiten und anschließend wieder einchecken. Eine kleinere Anpassung zum Test – zum Beispiel in der index.html – genügt. Mit:

```
git pull <repository-url-die-von-create-app-angezeigt-wird>
```

können wir den Quelltext abrufen und nach dem Bearbeiten mit:

```
git add *
git commit
git push
```

einfach wieder einchecken. Abbildung 1 veranschaulicht den Ablauf.

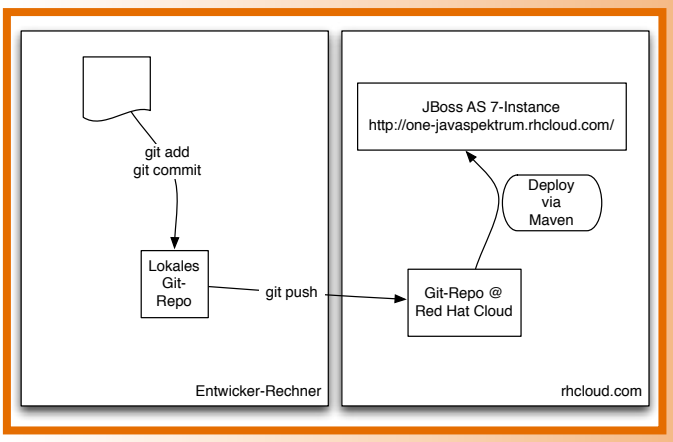

Abb. 1: Deploy in die Cloud

5.) Projektseite besuchen und staunen

Nach dem Check-in dauert es einen kurzen Moment und die eigene App läuft in der Cloud. Wir können die URL unserer Anwendung im Browser öffnen. Wer die URL nicht mehr weiß, kann im Console Panel auf der Weboberfläche nachsehen.

OpenShift und der Praxistest

Aus Interesse, inwieweit OpenShift auch mit Anwendungen klarkommt, die nicht von Red Hat ausgewählt wurden, habe ich im Web nach einer beliebigen Java EE-Anwendung gesucht, die man einfach als Anwendung herunterladen kann. Fündig wurde ich mit XWiki, einem Java-basierten Wiki.

In den Installationshinweisen zu XWiki steht sinngemäß: „Die Installation müsste einfacher sein.“ – Klingt, als haben wir einen Kandidaten gefunden, der nicht für die Cloud gemacht ist. Das Ergebnis sehen Sie auf [onejs].

Alltag mit OpenShift

Im Programmieralltag ist OpenShift unaufdringlich und stets souverän im Hintergrund. Dank Git ist die Einbettung in den Entwicklungsprozess völlig natürlich. Mit einem Push in den Versionsstrang der RedHat-Cloud stößt man diese an, den aktuell eingecheckten Softwarestand abzurufen und ihn in der Cloud zu bauen und zu installieren.

OpenShift setzt auf Maven: Beim Anlegen eines Java-Projekts mittels **`rhc-create-app`** wird unter anderem eine Datei pom.xml (Project Object Model) angelegt, in der wir die notwendigen Anpassungen vornehmen können. Wenn eine pom.xml vorhanden ist, wird in der RedHat-Cloud automatisch per Maven gebaut und deployt, sobald wir Änderungen ins OpenShift-Repository pushen.

Ein alternativer Weg steht ebenfalls zur Verfügung: Mithilfe sogenannter Marker-Dateien kann man der Cloud Kommandos schicken. Kopiert man zum Beispiel eine EAR- oder WAR-Datei blubb.war ins Unterverzeichnis deployments/ des Projektverzeichnisses und legt dann parallel dazu eine Datei blubb.war.dodeploy an und pusht diese per git zur Cloud-Plattform, kommt man auch ohne pom.xml und ohne Maven-Build aus. – Dieser Ansatz ist allerdings nur für kleine Projekte praktikabel. Stand November 2011 ist der Plattenplatz je Anwendung (einschließlich Git-Repository) nämlich – laut Dan McPherson von Red Hat – auf 500 MB begrenzt. Wir sollten daher den exploded mode des App-Servers verwenden, um so Plattenplatz zu sparen.

Ist OpenShift produktionstauglich?

Die Plattform ist – wie viele dieser Systeme – noch in der Entwicklung. Für Enterprise-Java-Software bietet sie allerdings den JBoss AS 7 als Laufzeit-Umgebung an, der voraussichtlich der kommende für den kommerziellen Einsatz unterstützte JBoss sein wird.

Die Voraussetzungen für einen produktiven Einsatz scheinen daher recht gut: Red Hat Enterprise (oder CentOS) als Betriebssystem, eine Java Runtime unseres Vertrauens und ein produktionstauglicher JBoss sind zumindest passende Zutaten.

Abgesehen von der Zuverlässigkeit der Laufzeitumgebung stellt sich mir die Frage nach der Zuverlässigkeit des Konfigurationsmanagements: Ziel des Konfigurationsmanagements ist das reproduzierbare Zusammenstellen von Komponenten und Diensten zu einem Gesamtsystem. Wenn dies von der OpenShift-Plattform mittels Maven gehandhabt wird, hat man neben den Vorkehrungen für einen reproduzierbaren Maven-Build auch noch die Unwägbarkeiten zu beherrschen, welche konkrete JBoss- und Maven-Version verwendet wurde. Welche Maven-Repositories herangezogen werden, ist dabei wohl die geringste Sorge.

Für die Cloud programmieren

Wenn man sich an die Spielregeln hält, die für die Entwicklung von Enterprise-Java-Systemen gelten, ist man fast auf der sicheren Seite. Leider genügt es nicht, die Finger vom Thread-Handling, der Synchronisation und der Parallelisierung zu lassen sowie auf das Halten von Zustand in statischen Attributen zu verzichten. Die PaaS-Anbieter schränken in der Regel die Technologie-Auswahl auf eine Menge von unterstützten Applikationsservern, Datenbankmanagementsystemen und Ähnlichem ein.

Wir können daher beispielsweise von der Beschränkung auf die interoperable Teilmenge der SQL-Elemente profitieren, die von „allen“ Datenbanksystemen unterstützt werden.

Fazit

Wer mit dem Kleingedruckten gut leben kann, kann seine Java EE-Anwendung bei einem Plattform-Anbieter kostengünstig (und ggf. auch gebührenfrei) betreiben lassen.

Dass die Zeit dafür reif zu sein scheint, zeigt unter anderem auch das ernorme Investitionsvolumen, mit dem derzeit „Platform as a Service“-Angebote erstellt, erweitert und vermarktet werden. Dass über kurz oder lang nicht alle Anbieter durchhalten werden, kann uns dank Java EE einigermaßen gleich sein, solange wir es schaffen, von der konkreten Plattform unabhängige Systeme zu entwickeln.

Dass die Applikationsserver in den letzten Jahren immer interoperabler wurden, spielt uns dabei weiter in die Hände. Für Projekte und Systeme, in denen Übertragbarkeit eine wesentliche Rolle spielt, bieten die Cloud-Plattformen einen geeigneten Einstieg an, in von Dritten betriebenen Umgebungen die Laufähigkeit der Software zu überprüfen.

Verschiedene Anbieter werben mit der Skalierbarkeit und vor allem Elastizität der Plattform. Inwieweit die hier vorgestellte Plattform OpenShift Express diese Merkmale erfüllt, muss sich in der Praxis erst beweisen. Dennoch sind das Deployment und der Test in der Cloud ein lohnendes (und vermutlich qualitätssteigerndes) Unterfangen.

Wie finde ich einen zukunftssicheren Plattform-Anbieter?

Ich bin dankbar für jede hilfreiche Idee. Meine Fähigkeit, die Zukunft vorherzusehen, war schon immer schlecht ausgeprägt. Aber in Zeiten, in denen Unternehmen schneller gekauft werden, als Lucky Luke seinen Revolver ziehen kann, ist nicht einmal die Zukunft von gesunden Unternehmen gewiss.

Als Kriterium für eine Cloud-Plattform kann sicherlich ihr wirtschaftlicher Erfolg herhalten. Wenn es viele zahlende Kunden gibt, ist die Chance gut, dass der Dienst für die Kunden ohne Anpassungen verwendet werden kann, selbst wenn die Plattform erweitert wird.

Die bloße Tatsache, dass eine Plattform verbreitet und bekannt ist, schützt nicht zwingend. Sicherlich wird weiterhin ein Dienst unter gleichem Namen bestehen, aber die Rückwärtskompatibilität ist nicht bei allen auf der Prioritätenliste weit oben vertreten.

Im Idealfall sollte die von uns entwickelte Software daher ausschließlich auf vom Plattform-Anbieter unabhängigen Standards basieren, um die Portierbarkeit zu gewährleisten.

Links

[gevap] G. Perrys Blog, Java PaaS, 17.10.2011, `http://gevaperry.typepad.com/main/2011/10/java-paas.html`

[git] `http://www.git-scm.com/`

[JSR342] Java Specification Requests 342: Java Platform, Enterprise Edition 7 (Java EE 7) Specification, `http://jcp.org/en/jsr/detail?id=342`

[onejs] Cloud-Anwendung des Autors mit XWiki zum Artikel, `http://one-javaspektrum.rhcloud.com/`

[rhc] OpenShift, Red Hat Cloud, `https://openshift.redhat.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