JAVA Mag inkl. DVD

W-JAX Challenge: Blick hinter die Kulissen »16

Deutschland €12,90 Österreich €14,80 Schweiz sFr 25,50 4.2010

Java™ magazin

EXTRA AUF DVD

1000 SEITEN JAVA-WISSEN

Über 150 Artikel aus 2009 • Mit intelligenter Suche

NOCH MEHR BONUS-MATERIAL AUF DVD:

VIDEOS • BÜCHER • IDEs • TOOLS

4 194586 712900 04

Datenträger enthält **Info- und Lehrprogramme** gemäß §14 JuSchG

JAVA Mag inkl. DVD

W-JAX Challenge: Blick hinter die Kulissen »16

S&S

Deutschland €12,90 Österreich €14,80 Schweiz sFr 25,50 4.2010

Java magazin

Java • Architekturen • SOA • Agile

www.javamagazin.de

DVD-INHALT

JAHRESARCHIV 2009

Über 150 Artikel auf DVD

Keynotes von Martin Odersky und Adam Bien

jax TV

Videos von der W-JAX 2009 in voller Länge

HIGHLIGHT

Kapitel aus drei Java-Büchern

Das große IDE-Paket

WEITERE INHALTE

Roma Meta Framework

Metawidget

GeoServer

uvm.

Alle DVD-Infos ab Seite 2

Interview mit Graeme Rocher

Alles zum Thema Grails 1.2, die Community und Roadmap 2010 »12

Deployit

Ein Tool-Radar zum Thema Deployment Automation »26

jax 2010

Alle Infos im Heft

ALLES GEO

Geodaten prozessieren mit GeoTools und WPS »37

Hands-on: Das Geo-Tutorial »28

Das Java Magazin Tutorial

Java EE 6 ausgepackt

Java Persistence API 2.0 »69

JavaFX

Reif für Unternehmensanwendungen »63

Flucht aus der „Jar-Hölle“

OSGi-Komponentenmodell »82

4 194586 712900 04

Datenträger enthält Info- und Lehrprogramme gemäß §14 JuSchG

Elastic Compute Cloud: Darf es noch ein Server mehr sein?

Die Elastic Compute Cloud (EC2) von Amazon erlaubt die Nutzung von Compute/Network/Storage ferngesteuert über Web Services. Wir werden vertiefen, wie die Steuerung von EC2 über Web Services und über die Kommandozeile funktioniert. In einer Application Tier werden wir Tomcat-Instanzen hinter einem Load Balancer anlegen und die Zahl der Instanzen automatisch an die Last anpassen (Auto Scaling und Elastic Load Balancing). Und als Database Tier werden wir eine Oracle-Datenbank und einen Relational Database Service (Automatisiertes Backup- und Patch-Management) aufsetzen.

von Lothar Wieske

JAX Speaker

Mit der Elastic Compute Cloud (EC2) hatten wir im letzten Java Magazin Infrastructure-as-a-Service in der Public Cloud von Amazon kennen gelernt [1]. Wir hatten die zugrunde liegenden Konzepte und Ressourcen von EC2 theoretisch angeschaut und mit dem Start einer Instanz über die grafische Konsole auch praktisch eingesetzt. In diesem Artikel wollen wir nun Instanzen über die Kommandozeile starten.

Artikelserie

Teil 1: Amazon Web Services: Flaggschiff des Cloud Computings

Teil 2: Elastic Compute Cloud: Darf es noch ein Server mehr sein?

Teil 3: BASE: Internetanwendungen jenseits der Standards

AWS Security Credentials

Die Nutzung von Amazon Web Services wird durch drei Gruppen von Security Credentials abgesichert:

- Sign-in Credentials
- Access Credentials
- Account Identifiers

Die Sign-in Credentials als Paar aus E-Mail-Adresse und Password sichern den AWS Account. Beide hatten wir im letzten Artikel zur Anmeldung bei der AWS-Konsole verwendet. Damit konnten wir den Status unserer Instanzen und die aufgelaufenen Kosten überwachen, aber auch die sonstigen Security Credentials einsehen. Für das Password ist aufgrund seiner Mächtigkeit strengste Geheimhaltung angesagt und seinem Schutz sollte entsprechende Aufmerksamkeit gewidmet werden (Wahl, Wechsel). Gerade weil die Kombination aus E-Mail-Adresse und Passwort so mächtig ist, konnten wir im Beispiel unsere EC2-Instanzen ohne tiefergehende Konfiguration für die Security Credentials starten. (Die Arbeit mit den API-Tools für EC2, CloudWatch, Auto Scaling, Elastic Load Balancing und RDS gelingt erst nach erfolgreicher Konfiguration der spezifischeren Access Credentials.) Amazon bietet Ihnen als zusätzliche Sicherung für den Zugang über Sign-in Credentials einen Multi-Factor-Authentication-Code. Dazu bestellen Sie ein entsprechendes Gerät bei Gemalto, das Ihnen in

halbminütigem Wechsel eine sechsstellige Zahlenfolge anzeigt. Wenn Sie dann für Amazon Web Services die Multi-Factor Authentication aktivieren, wird von Ihnen bei der Anmeldung zusätzlich zu E-Mail-Adresse und Password dieser Multi-Factor-Authentication-Code abgefragt.

Die Access Credentials sind ihrerseits wieder in drei Untergruppen gegliedert: Access Keys, X.509 Certificates und Key Pairs. Access Keys (Abb. 1) bestehen aus Access Key ID und Secret Access Key. Die Access Key ID ist eine zwanzigstellige alphanumerische Zeichenfolge, beispielsweise AKIAJFA643ICQZQEFW7A – in bestimmten Fällen ist die Access Key ID für andere sichtbar. Der Secret Access Key ist eine vierzigstellige alphanumerische Zeichenfolge mit Schrägstrichen und Pluszeichen, z. B. JIKuuIDKa2qtcHJPGq/NrLJxVuUhr6KVmk9R8JaG – der Secret Access Key ist nie für andere sichtbar und geht nur Sie etwas an. Strengste Geheimhaltung ist für den Secret Access Key genauso angesagt wie für das Passwort. Sie können für Ihren Account jeweils zwei Access Keys erzeugen und somit regelmäßig neue erstellen und durchrotieren.

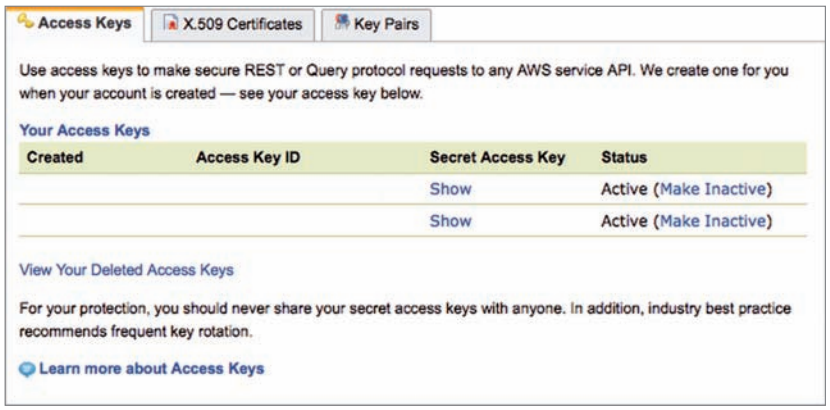

Abb. 1: Access Keys

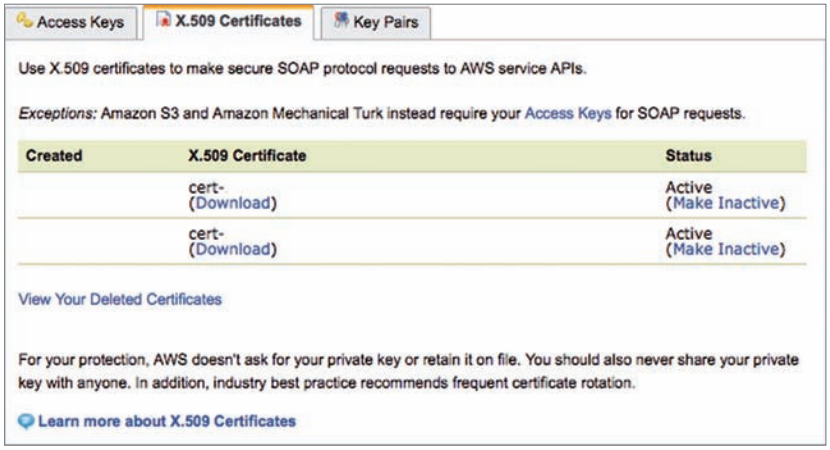

Abb. 2: X.509 Certificates

X.509 Certificate und Private Key gehören zu den X.509 Certificates. Sie sind kleine Dateien mit langen Namen der Gestalt *cert-VFFZUMMGFZ45FGUJBFE57IJBVFQKPSYG.pem* und *pk-VFFZUMMGFZ45FGUJBFE57IJBVFQKPSYG.pem*. Beide verwenden Sie für das SOAP API von Amazon Web Services, aber auch für die Absicherung der von Ihnen erzeugten Amazon Machine Images. Natürlich sollten Sie den X.509 Private Key geheim halten. Sie dürfen ihn auch nicht verlieren, denn Amazon bewahrt nur eine Kopie des X.509 Certificate zur Gegenprüfung Ihrer SOAP Requests auf. Sie können für Ihren Account jeweils zwei X.509 Certificates erzeugen und somit regelmäßig neue erstellen und durchrotieren.

Ein Key Pair ordnet einem Namen jeweils einen Public SSH Key und Private SSH Key zu. Mit einem Key Pair wird sichergestellt, dass nur Sie sich

auf einer gestarteten Instanz anmelden können. Deswegen geben Sie den Namen des gewünschten Key Pairs beim Starten der Instanz an. EC2 kopiert den Public SSH Key in die Instanz und Sie geben beim Aufruf von SSH ihren Private SSH Key an. Und so gelingt die Authentifizierung für den *root*-Benutzer Ihrer Instanz ohne Kennwort. (Für die Ubuntu AMIs von Canonical ist eine Besonderheit zu beachten: Hier verbinden Sie sich mit dem Ubuntu-Benutzer.) Eine weitergehende Unterstützung für Identity und Acces Management auf den gestarteten Instanzen gibt es (leider) nicht.

Bleibt noch die Gruppe der Account Identifiers mit AWS Account ID und Canonical User ID, die zum Einsatz kommen, wenn AWS-Ressourcen durch mehrere Accounts gemeinsam genutzt werden sollen. Dabei bedient die AWS Account ID eine Vielzahl von AWS-Ressourcen wie EC2 AMIs und EBS Snapshots. Die Canonical User ID findet ausschließlich Eingang in die Access Control Lists für S3 Buckets und Objects. Die AWS Account ID besteht aus einer zwölfstelligen Ziffernfolge, die mit oder ohne Bindestriche verwendet wird: 1234-5678-1234 bzw. 123456781234. Die Canonical User ID besteht aus einer 64-stelligen alphanumerischen Zeichenfolge.

Amazon Web Services – Steuerung mit Query/REST/SOAP

Natürlich lässt sich die Elastic Compute Cloud (EC2) über die Konsole bequem steuern. Für eine weitergehende Automatisierung benötigen wir aber eine oder mehrere Programmierschnittstellen zur Einbindung in Skripte und Werkzeuge. Die Amazon Web Services (AWS) bieten insgesamt drei verschiedene Schnittstellentypen an, die als Query, SOAP und REST bezeichnet werden. Tabelle 1 zeigt die Zuordnung und Verfügbarkeit der Schnittstellen für die von uns genutzten Infrastructure Services.

Service	Query	REST	SOAP
CloudFront		✓	
EC2	✓		✓
RDS	✓		✓
S3		✓	✓

Tabelle 1: Query/REST/SOAP Interfaces für Amazon Web Services

AWS Query Interfaces

Das AWS Query Interface bildet seine Operationen auf HTTP Requests mit einfachen Name/Wertpaaren als Parameter ab. Beim Query Interface werden die Parameter entweder in der URI eines GET Requests oder im Body eines POST Requests übertragen. Die Method-Komponente des HTTP Requests (GET/POST) zeigt lediglich an, wo die Parameter zu finden sind.

Die Authentifizierung beim Query Interface erfordert vier Schritte. Zunächst stellen Sie den Request mit seinen Parametern zusammen und überführen diese Zeichenkette in ein standardisiertes Format. Dann errechnen Sie für die Zeichenkette mit dem Secret Access Key eine Signature (RFC 2104-compliant HMAC-SHA1 oder HMAC-SHA256 Hashcode/Base64 Encoding). Nun beinhaltet der Request insbesondere die Parameternamen *AccessKeyID*, *SignatureMethod* (*HmacSHA1* bzw. *HmacSHA256*) und *Signature* mit den entsprechenden Werten. Diesen Request übertragen Sie per HTTPS an den AWS Endpoint. Dort holt sich Amazon den zu Ihrer Access Key ID gehörenden Secret Access Key und berechnet gleichfalls eine Signature nach der ausgewählten *SignatureMethod* für Ihren Requests. Bei Übereinstimmung beider Signaturen ist Ihr Request authentifiziert.

AWS REST Interfaces

Das AWS REST Interface verwendet für seine Operationen die Standardelemente eines HTTP Requests. Die Method-Komponente des HTTP Requests (DELETE/GET/PUT/POST) zeigt an, was die Operation machen wird, die URI-Komponente bezeichnet die Ressource(n), mit denen gearbeitet wird, und die Body-Komponente übermittelt gegebenenfalls Daten, mit denen gearbeitet wird.

Die Authentifizierung beim REST Interface funktioniert in vier Schritten. Zunächst stellen Sie einen Request zusammen, der alle Informationen zur Operation enthält, und überführen diese Zeichenkette in ein standardisiertes Format. Dann errechnen Sie für diese Zeichenkette mit dem Secret Access Key eine Signature (RFC 2104-compliant HMAC-SHA1 Hashcode/Base64 Encoding). Nun erstellen Sie für den Request einen HTTP Authorization Header und binden darin Ihre Access Key ID und die Signature ein (Authorization: *AWS <AWSAccessKeyId>:<Signature>*). Mit diesem HTTP Authorization Header übertragen Sie Ihren Request per HTTPS an den AWS Endpoint. Dort holt sich Amazon den zu Ihrer Access Key ID gehörenden Secret Access Key und berechnet gleichfalls die Signature für Ihren Requests. Wenn's passt, ist Ihr Request authentifiziert.

AWS SOAP Interfaces

Beim AWS SOAP Interface bildet ein XML-Dokument die gewünschten Operationen mit allen Informationen vollständig ab. Das HTTP-Protokoll ist von den eigentlichen Operationen entkoppelt und fungiert nur als Trägerschicht für die Kommunikation.

Auch bei der Authentifizierung für das SOAP Interface sind vier Schritte erforderlich. Zunächst wird der eigentliche Request als XML-Dokument für den SOAP Body des umfassenden SOAP Envelope formuliert, entsprechend den Vorgaben des WSDL/XML Schema Documents für Request- und Response-Strukturen [2], [3], [4]. Zur Absicherung wird für den SOAP Envelope ein SOAP Header nach den Vorgaben der WS-Security-Standards mit den drei zentralen Elementen *Binary Security Token*, *Signature* und *Timestamp* gebaut. Im *Binary-Security-Token*-Element wird das X.509 Certificate verbaut, in der *Signature* wird der eigentliche Signature-Code mit der *SignatureMethod* untergebracht und die *Timestamp* ermöglicht die Absicherung gegen Replay-Attacken (Timestamp und Expires gibt es auch für Query und Rest). Diesen SOAP Envelope übertragen Sie per HTTPS an den AWS Endpoint. Dort holt sich Amazon den zu Ihrer Access Key ID gehörenden Secret Access Key und berechnet gleichfalls die Signature

für Ihren Requests. Wenn's passt, ist Ihr Request authentifiziert.

AWS APIs und Security Credentials

Wahrscheinlich werden Sie als Nutzer der Web Services keinen der Query, REST oder SOAP Requests selbst zusammenbauen, sondern je nach Programmiersprache und Technologieplattform unter den zahlreichen verfügbaren Bibliotheken und/oder Werkzeugen auswählen. Für Query Interface und REST Interface benötigen Sie also Access Key ID und Secret Access Key, während für das SOAP Interface X.509 Certificate und X.509 Private Key zum Einsatz kommen. Mit dem Wissen um diese Zuordnung und der jeweiligen Gestalt und Bedeutung der Security Credentials gelingt Ihnen die Konfiguration der API-Tools vielleicht etwas leichter.

Oracle-Datenbank mit EC2

Die Installation einer Oracle-Datenbank-11g-Enterprise-Edition erledigt sich innerhalb der Elastic Compute Cloud schnell und einfach. Oracle stellt nämlich ein Amazon Machine Image bereit, auf dem die Datenbanksoftware bereits installiert ist. Wir starten also eine Instanz, konfigurieren Network und Storage und erzeugen dann mit dem Installationsskript für die Datenbank die Tablespaces. Wir werden über die Kommandozeile arbeiten und damit die grundlegenden Ressourcen (Image, Instance, Key Pair, Security Group, Elastic Block Storage) von EC2 im Zusammenspiel kennen lernen.

Zunächst installieren wir die EC2-API-Tools. Wir entpacken die zugehörige ZIP-Datei an einem Ort unserer Wahl und setzen einige Umgebungsvariablen. Zum einen muss *JAVA_HOME* gesetzt sein – die EC2-API-Tools sind in Java implementiert und fordern es so. *EC2_HOME* wird auf den Verzeichnisnamen mit der entpackten ZIP-Datei gesetzt und *EC2_HOME/bin* in die Umgebungsvariable *PATH* eingebunden. Und dann gilt es noch, *EC2_CERT* und *EC2_PRIVATE_KEY* auf die absoluten Pfadnamen Ihrer *cert-*.pem-* und *pk-*.pem*-Dateien zu setzen. Ob alles funktioniert hat, können wir mit dem *ec2-describe-regions*-Befehl testen.

Für den entscheidenden *ec2-run-instances*-Befehl benötigen wir ein Key Pair und eine Security Group, deswegen erzeugen wir beide vorab:

> CloudFront, EC2, S3 und RDS sind jeweils eigenständige Dienste mit API-Tools.

```
$ ec2-add-keypair jm042010ora
$ ec2-add-group jm042010ora --description
                                    "Oracle DB"
```

Der *ec2-add-keypair*-Befehl liefert als Ausgabe den neu erzeugten RSA Private Key. Den zugehörigen Public Key bewahrt Amazon auf, den Private Key jedoch nicht. Deswegen packen wir den Inhalt dieser Ausgabe zwischen den *BEGIN*- und *END*-Marken in eine Datei mit dem Namen *jm042010ora.pem* und schützen diese vor Lese- und Schreibzugriffen durch *group* und *others*:

```
$ ec2-run-instances ami-7ecb2f17 --key jm042010ora
                                    --group jm042010ora
                          --instance-type m1.large
```

Jetzt läuft eine Instanz mit vorinstalliertem Oracle Database 11g auf Oracle Enterprise Linux 5 mit 64-bit CPU (4 EC2 Compute Units) und 7.5 GB RAM Hauptspeicher. Für die Netzwerkkonfiguration verschaffen wir uns eine Elastic-IP-Adresse und weisen sie der Instanz zu. Dann öffnen wir die Security Group für den Zugriff durch SSH, damit wir uns anmelden können:

```
$ ec2-allocate-address
$ ec2-associate-address 204.236.226.24 --instance
                                    i-f8477390

$ ec2-authorize jm042010ora --protocol tcp
                                    –port-range 22
                          --source-subnet 84.177.120.201/32
```

Einige Befehle erzeugen Bezeichner und Werte, die wir in nachfolgenden Befehlen verwenden. Damit sich diese Beschreibung etwas flüssiger liest, sind die Angaben aus einem Beispieldurchlauf in den Befehlen verblieben. Sie müssen hier natürlich Ihre eigenen Angaben sammeln und einsetzen: Instanzbezeichner (anstelle i-f8477390), Elastic IP (anstelle 204.236.226.24, Volume (anstelle vol-3427e15d), Ihre IP-Adresse (anstelle 84.177.120.201). Jetzt erzeugen wir ein EBS Volume der Größe 10 GB und weisen es der Instanz zu:

```
$ ec2-create-volume --size 10
--availability-zone
                                    us-east-1c
$ ec2-attach-volume vol-3427e15d --instance
                                    i-f8477390 --device /dev/sdf
```

Mit der Zuweisung des Volumes *vol-3427e15d* zum Device */dev/sdf* schaffen wir die Grundlage für seine spätere Nutzung innerhalb der Instanz, d. h. die Befehle *mkfs* und *mount*. Dann melden wir uns erstmalig bei der Instanz als Benutzer *root* an. Zunächst läuft ein Skript, mit dem die Lizenzbedingungen angezeigt werden. Diese Lizenz gilt es zu lesen und anzunehmen. Dann verneinen wir das Angebot, jetzt eine Datenbank anzulegen. Abschließend vergeben wir ein Password für den Benutzer *oracle*. Nun können wir auf dem EBS Volume (sprich: */dev/sdf*) ein *ext3*-Dateisystem einrichten:

```
$ ssh -i ~/.ec2/oracle.pem root@ec2-204-236-226-
                                    24.compute-1.amazonaws.com

> mkfs.ext3 /dev/sdf
```

Vor dem Anlegen der Datenbank müssen wir in der Datei *~oracle/.bash_profile* die Variable *PUBLIC_HOSTNAME* auf den richtigen Wert setzen (*ec2-204-236-226-24.compute-1.amazonaws.com*) – diese Korrektur ist notwendig, weil die Elastic IP erst nach dem Starten der Instanz zugewiesen wird und wir für Enterprise Manager und Application Express den DNS-Namen für die Elastic IP benötigen. Nun starten wir das handelsübliche Skript für das Anlegen der Datenbank: */home/oracle/scripts/run_dbca.sh*.

Für die Eingabeaufforderungen wählen wir */dev/sdf* als Disk Device für Datenbank und Flash Recovery Area und */u02* als Mount Point. Nach dem Namen für die Datenbank werden Kennworte abgefragt. Die abschließenden Zeilen des Installationsskripts zeigen, unter welchen URLs der Enterprise Manager und Application Express zu erreichen sind. Wir können die *ALLOW*-Regeln für Enterprise Manager (Port 1158) und Application Express (8080) in der Security Group *jm042010ora* jetzt hinzufügen und sie werden dann in wenigen Sekunden aktiv:

```
$ ec2-authorize jm042010ora --protocol tcp
                            --port-range 1158
        --source-subnet 84.177.120.201/32
$ ec2-authorize jm042010ora --protocol tcp
                            --port-range 8080
        --source-subnet 84.177.120.201/32
```

Fertig. Jetzt können wir Enterprise Manager mit der Kennung *SYSTEM* und Application Express mit der Kennung *ADMIN* verwenden.

Application Tier mit Tomcat und Elastizität

Für die Application Tier starten wir mit einem neuen Key Pair und verfahren wie oben: *ec2-add-keypair jm042010app*. Mit diesem Key Pair starten wir eine Instanz mit dem Amazon Machine Image *ami-2eb05347*. Mit dem Starten der Instanz wird der installierte Tomcat Server als Dienst gleich mit gestartet. Die Security Group *default* sollte die Ports 80 und 8080 als Allowed Connections enthalten (Listing 1).

Jetzt müssen wir Last auf diesen Server geben, um grundsätzliche Kennzahlen für den Durchsatz zu bekommen. Als Werkzeug für die Lastgenerierung eignet sich Apache Benchmark als Bestandteil einer Apache-HTTPD-Installation; es wird einfach mit dem Befehl *ab* aufgerufen. Für *ab* verschaffen wir uns nun eine Kennzahl für die Zahl der Requests pro Minute in einem einzelnen Thread. Weil hier unterschiedlichste Parameter (CPU-Leistung, Netzwerkanbindung) zu berücksichtigen sind, kann dieser Artikel keine für Sie gültigen Angaben liefern. In der Konfiguration des Autors lieferte der Aufruf des *HelloWorldServlet* mit 600 aufeinanderfolgenden Requests (-n 600) für einen einzelnen Thread (-c 1), etwa 3-4 Requests pro Sekunde und eine Gesamtlaufzeit von ca. 3 Minuten: *$ ab -n 600 -c 1 http://<<DNS>>/servlets-examples/servlet/HelloWorldExample*.

Nach diesen Vorbereitungen widmen wir uns CloudWatch, Elastic Load Balancing und Auto Scaling, die im Verbund eine automatisierte Lastanpassung ganzer Gruppen von Instanzen erlauben.

Mit CloudWatch kann für EC2-Instanzen ein Monitoring an- und ausgeschaltet werden – auch nach dem Start des Servers. Damit werden für die Instanz grundlegende Leistungsdaten zu CPU, Network und Storage gesammelt. Die gesammelten Daten sind über Web Services zugreifbar und werden bis zu zwei Wochen aufbewahrt – auch nach dem Stoppen der Instanz.

Auto Scaling fährt EC2-Kapazitäten (Anzahl der Server) automatisch nach oben oder nach unten – nach Regeln und Bedingungen, die Sie auf der Grundlage der aktuellen Leistungsdaten Ihrer laufenden Instanzen festlegen. Für die Verfügbarkeit der aktuellen Leistungsdaten setzt Auto Scaling von daher Cloud Watch voraus. Mit Auto Scaling können z. B. mittägliche Lastspitzen mit zusätzlichen Servern ebenso wie nächtliche Ruhepausen mit freigegebenen Servern behandelt werden. Automatisierte Versorgung mit der jeweils richtigen Anzahl von Servern, das ist wahrlich ein Traum für alle Suchenden nach Stellschrauben für die Optimierung von Betriebskosten – aber die Architektur der Anwendung muss diese dynamische Skalierung im laufenden Betrieb auch hergeben. Das nimmt Auto Scaling den Architekten nicht ab.

Elastic Load Balancing verteilt Anwendungslasten automatisch auf Ihre Instanzen in einem entsprechenden Pool. Zusätzlich kann es auch schwergängige oder ausgefallene Instanzen erkennen und leitet die Anfragen in der Folge geeignet um. Elastic Load Balancing können Sie für eine einzelne Availability Zone einrichten oder auch mehrere Availability Zones einbeziehen, um die Möglichkeiten noch besser auszureizen. Wir werden als einfaches Beispiel für den Einstieg innerhalb einer Availability Zone einen Elastic Load Balancer aufsetzen, hinter dem Auto Scaling Group mit mindestens zwei und höchstens vier Tomcat-Instanzen liegt. Wir werden dann mit dem Apache Benchmark *ab* Lasten generieren und das Auslösen der Trigger überwachen.

Zunächst müssen die Werkzeuge für die Kommandozeilensteuerung von CloudWatch, Auto Scaling und Load Balancing installiert und konfiguriert werden. Dafür gilt das oben beschriebene Strickmuster, die Umgebungsvariablen für die entpackten ZIP-Dateien sind: *AWS_CLOUDWATCH_HOME*, *AWS_AUTO_SCALING_GROUP* und *AWS_ELB_HOME* [6], [7], [8]. Wir legen den Elastic Load Balancer an:

```
$ elb-create-lb JM042010ELB –headers
      --listener "lb-port=80,
            instance-port=8080,protocol=HTTP"
      --availability-zones us-east-1c
```

In der Ausgabe zu diesem Befehl finden wir den DNS-Namen des Elastic Load Balancers in der Bauart *JM042010ELB-*.us-east-1.elb.amazonaws.com*. Nun geht es an die Launch-Konfiguration, damit Auto Scaling auch neue Instanzen ins Spiel bringen kann:

```
$ as-create-launch-config JM042010LC
        --image-id ami-2eb05347
        --instance-type m1.small
        --key jm042010app
        --group jm042010app
```

Nun erzeugen wir eine Auto Scaling Group und verknüpfen sie mit dem Load Balancer und der Launch Configuration. Außerdem legen wir die minimale und maximale Anzahl von Tomcat-Instanzen

Listing 1

```
$ ec2-add-group jm042010app --description “Tomcat“
$ ec2-authorize jm042010app --protocol tcp --port-range 80
        --source-subnet 84.177.120.201/32
$ ec2-authorize jm042010app --protocol tcp --port-range 8080
        --source-subnet 84.177.120.201/32
$ ec2-run-instances ami-2eb05347 --key jm042010app
                                 --group jm042010app
        --instance-type m1.large
```

fest. Einstweilen setzen wir beide Zahlen auf null. Wir können später mit *as-update-auto-scaling-group* nachsteuern:

```
$ as-create-auto-scaling-group JM042010SG
        --load-balancers JM042010ELB
        --launch-configuration JM042010LC
        --min-size 0 --max-size 0
        --availability-zones us-east-1c
```

Jetzt noch den Trigger zum Starten neuer Server hinzufügen und unsere elastische Application Tier mit Tomcat-Instanzen ist vollständig konfiguriert .

```
$ as-create-or-update-trigger JM042010Trigger
        --auto-scaling-group JM042010SG
        --namespace="AWS/EC2"
        --measure NetworkOut
        --statistic Average
        --dimensions "AutoScalingGroupName=
                                JM042010SG"
        --period 60
        --unit Bytes
        --lower-threshold 90000
        --upper-threshold 120000
        --lower-breach-increment=-1
        --upper-breach-increment 1
        --breach-duration 300
```

Jetzt können wir diese Konfiguration schrittweise in Betrieb nehmen. Zuvor ist vielleicht noch ein Blick in die Dokumentation von CloudWatch, Auto Scaling und Elastic Load Balancing angeraten. Denn Automatisierung heißt hier neben Arbeitserleichterung auch Kostenerzeugung. Die folgenden drei Aktivitäten sollten Sie beherrschen, um im Fall des Falles die Reißleine ziehen zu können: Überwachen des Auto Scaling (*as-describe-auto-scaling-groups*, *as-describe-auto-scaling-activities*), Herunterfahren des Auto Scaling (*as-update-auto-scaling-group* speziell mit *min-size 0* und *max-size 0*) und Löschen der Konfiguration und seiner Elemente (*as-delete-auto-scaling-group*, *as-delete-launch-configuration*, *as-delete-trigger*, *elb-delete-lb*). Im Folgenden werden drei Schritte zum Arbeiten und Kennenlernen dieser Konfiguration erläutert:

1. Zunächst können wir automatisiert eine Tomcat-Instanz hochfahren, indem wir *min-size* und *max-size* der Auto Scaling Group *JM2010SG* auf 1 setzen. Anschließend sollten wir die Konfiguration des Load Balancers überprüfen, indem wir ihm einen entsprechenden Request für das *HelloWorldServlet* senden.
2. Nun können wir in der Auto Scaling Group die *max-size* auf 2 setzen und anschließend einen Trigger auslösen. Dazu können wir den Concurrency-Parameter des Apache Benchmark erhöhen und auf 2 oder 3 setzen. Jetzt empfiehlt es sich, mit *mon-get-stats* die von CloudWatch für den ausgehenden Netzwerkverkehr gelieferten Leistungsdaten anzuschauen:

```
$ mon-get-stats NetworkOut
  --statistics Average
  --period 60
  --namespace "AWS/EC2"
  --dimensions "ImageId=ami-2eb05347"
```

3. Im dritten Schritt können wir nun den Lower Threshold und Upper Threshold des Triggers so einstellen, dass wir die Zahl der Tomcat-Instanzen mit einer entsprechenden Abfolge von *ab*-Kommandos stufenweise von 2 auf 4 hochfahren, dort halten und danach wieder stufenweise von 4 auf 2 herunterfahren können. Diese kleine Aufgabenstellung hört sich einfach an. Bei der Umsetzung bieten sich jedenfalls hinreichend Lernchancen zur Arbeitsweise von CloudWatch, Auto Scaling und Elastic Load Balancing.

AWS Relational Data Service

Mit dem Relational Database Service bietet Amazon eine relationale Datenbank (MySQL) als Managed Service an. Das RDS API startet eine MySQL-5.1-Instanz mit einer Auswahl aus fünf unterschiedlichen Instanzklassen und einer Storage-Größe zwischen 5 GB und 1 TB. Zwar kann sich der Nutzer nicht auf dieser Instanz anmelden, über MySQL-Netzwerkprotokoll und -Clientbibliotheken kann er jedoch in der gewohnten Weise mit der Datenbank arbeiten.

Management und Maintenance der RDS-Instanz übernimmt Amazon. Täglich öffnet sich ein zweistündiges Zeitfenster, in dem Amazon ein Backup für Ihre MySQL-Datenbank durchführt. Dieses Backup wird für eine bestimmte Anzahl von Tagen (Retention Period) aufbewahrt; innerhalb dieser Frist kann die Datenbank mit dem Befehl *restore-db-instance-to-point-in-time* zurückgesetzt werden. Wöchentlich öffnet sich ein vierstündiges Zeitfenster, in dem Amazon für die RDS-Instanz eventuelle MySQL Patches einspielt und von Ihnen aufgesetzte Skalierungen vornimmt. Sie können darin für Ihre Instanz mit dem Befehl *rds-modify-db-instance* die Instanzklasse ändern und/oder den Storage vergrößern lassen. Beide Zeitfenster sind durch den Benutzer konfigurierbar. Zusätzlich unterstützt Sie das RDS API mit einem Eventsystem, das mit dem Befehl *rds-describe-events* Hinweise auf etwaige Vorkommnisse meldet.

Erweiterungen für RDS sind bereits in der Pipeline. Zum einen wird das Konzept der Reserved Instances von EC2 auch auf RDS übertragen. Sie zahlen dann einen einmaligen Beitrag, der Ihnen für die späteren stündlichen Verbrauchskosten einen Abschlag sichert. (Insgesamt liegt der Preis für eine RDS-Instanz etwa 30 % über dem Preis für eine EC2-Instanz.) Zum anderen wird es mit synchron replizierten Instanzen in unterschiedlichen Availability Zones eine verbesserte Verfügbarkeit geben.

Die Amazon-RDS-Werkzeuge für die Kommandozeile funktionieren in der schon besprochenen Weise mit der Variablen *AWS_RDS_HOME*, arbeiten jedoch mit dem AWS Access Key und dem Secret Access Key für Ihren AWS Account. Dazu gibt es eine Datei *AWS_RDS_HOME/credential-file-path.template* mit folgendem Inhalt:

```
AWSAccessKeyId=<Write your AWS access ID>
AWSSecretKey=<Write your AWS secret key>
```

Wir kopieren diese Datei mit einem Namen unserer Wahl an einen Ort unserer Wahl. Wir tragen dann unsere Informationen ein und sichern die Datei gegen Zugriffe durch Dritte. Dann setzen wir die Umgebungsvariable *AWS_CREDENTIAL_FILE* auf den vollständigen Dateinamen dieser Datei. Nun legen wir eine RDS-Instanz *db.m1.large* mit einer Größe von 10 GB an:

```
$ rds-create-db-instance --db-instance-identifier
                                jm042010db
        --db-instance-class db.m1.large
```

```
--allocated-storage 10
--engine mysql5.1
--master-username    <<your choice>>
--master-user-password <<your choice>>
```

Wir können die Bereitstellung der Datenbank jederzeit mit dem Befehl *$ rds-describe-db-instances* ansehen. Nach dem Wechsel des Status von *creating* auf *available* können wir die Datenbank benutzen und sehen in der Ausgabe des Befehls den Servernamen und den Connection String für den Zugriff über GUI- oder CLI-Werkzeuge. Als letzten Schritt müssen wir noch die RDS Security Group für den eingehenden Netzwerkverkehr freischalten. Dafür können wir Verbindungen von einer oder mehreren EC2 Security Groups erlauben. Wir können für den Zugriff außerhalb von EC2 auch Verbindungen für spezifische IP-Adressen oder Adressbereiche erlauben (CIDR):

```
$ rds-authorize-db-security-group-ingress default
                                --cidr-ip 84.177.120.201/32.
```

Nun können wir mit der Datenbank arbeiten, beispielweise mit den CLI-Werkzeugen von MySQL oder den GUI-Werkzeugen der Eclipse Datatools. Wenn unsere Datenbank größer wird, können wir sie z. B. mit folgendem Befehl auf 20 GB vergrößern:

```
$ rds-modify-db-instance jm042010db --apply-
immediately -s 20.
```

Neben dem automatisierten Backup im täglichen Managementzeitfenster können wir bedarfsgetrieben auch Snapshots der Datenbank erstellen: *$ rds-create-db-snapshot jm042010db -s jm042010-backup-2010-01-15*. Sowohl aus einem solchen spontanen Snapshot als auch aus einem regulären Backup kann eine neue RDS-Instanz erzeugt werden. Mit dem regulären Backup kann die Datenbank auf jeden Zeitpunkt innerhalb der Retention Period zurückgesetzt werden, nur die letzten fünf Minuten vor dem jüngsten Backup sind nicht erreichbar. Damit ergeben sich verschiedene Einsatzmöglichkeiten im Issue- und Testmanagement, weil eine Datenbank für die Analyse eines Fehlerfalls oder mehrere Datenbanken für den Einsatz in Testszenarien mit wenigen Handgriffen über die Kommandozeile verfügbar werden. Beim tiefergehenden Verständnis der Möglichkeiten hilft die gut gemachte Dokumentation.

Jede RDS-Instanz exportiert Metriken an CloudWatch, z. B. CPU-Auslastung (Prozent), freien Storage Space (Bytes) und Database Connections (Anzahl). Abschließend löschen wir alle Ressourcen, die wir im Laufe dieses Artikels angelegt haben, damit keine weiteren Kosten entstehen. Wir haben nun Application Tier und Database Tier mit verschiedenen Instanzen ausgestaltet und dabei unterschiedliche Konzepte der Elastic Compute Cloud kennen gelernt.

Zusammenfassung

Jetzt schauen wir noch einmal nach den Essential Characteristics des Cloud Computings:

- *On-Demand Self-Service*: "A consumer can unilaterally provision com-

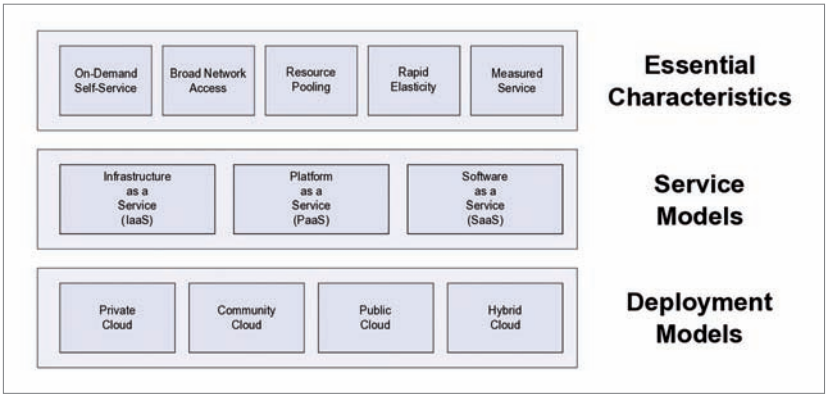

Abb. 3: NIST-Definition zu Cloud Computing

puting capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider."

- *Broad Network Access*: "Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms."
- *Resource Pooling*: "The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction … Examples of resources include storage, processing, memory, network bandwidth, and virtual machines."
- *Rapid Elasticity*: "Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time."
- *Measured Service*: "Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service ... Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service."

On-Demand Self-Service: Der Zugang über Web Services und die dahinter liegende technologische Unterstützung durch Standardisierung, Virtualisierung und Automatisierung mündet in Lights-Out- oder Zero-Touch-Rechenzentren mit riesigen Kapazitäten und deren Bereitstellung ohne Voranmeldung.

Broad Network Access: Die administrative und operative Nutzung der Instanzen in der Elastic Compute Cloud erfolgt ausschließlich über das Netzwerk mit standardisierten Protokollen und Bandbreiten von 250-1 000Mb/s je nach Instanzklasse.

Resource Pooling: Der AWS Account und die Authentifizierungsmechanismen der APIs vermitteln den Eindruck eines persönlichen Ressourcenvorrats im eigenen Rechenzentrum. („The Network is YOUR Computer.") Betrachtet man alle AWS Accounts zusammen, entsteht ein beeindruckendes Bild. Schätzungen gehen von insgesamt 15 Millionen gestarteten Instanzen seit dem Start und mehr als 50 000 gestarteten Instanzen pro Tag im Oktober 2009 aus [10].

Rapid Elasticity: Was da geht und wie es geht haben wir uns im Zusammenspiel von CloudWatch, Auto Scaling und Elastic Load Balancing angeschaut.

Measured Service: Messungen zur Nutzung der Elastic Compute Cloud finden in unterschiedlicher Hinsicht statt. Monetär wirksam werden die Messungen von Amazon über die monatliche Abrechnung. Mit CloudWatch kann der Nutzer selbst Leistungsdaten für die eigenen Ressourcen im laufenden Betrieb ziehen. Ein Dienst wie CloudStatus ermöglicht Überblicke und Rückblicke auf Kennzahlen der Elastic Compute Cloud im Allgemeinen [11]. ■

Lothar Wieske ist Enterprise Architect. Als Architekt hat er Anwendungslandschaften entworfen und als Coach hat er Teams bei Veränderungen begleitet. Seine Themen sind Cloud Computing, Model-driven Development sowie systemisches Coaching und agile Entwicklung.

Links & Literatur

[1] http://csrc.nist.gov/groups/SNS/cloud-computing/index.html
[2] http://ec2.amazonaws.com/doc/2009-11-30/AmazonEC2.wsdl
[3] https://rds.amazonaws.com/doc/2009-10-16/AmazonRDS.wsdl
[4] http://s3.amazonaws.com/doc/2006-03-01/AmazonS3.wsdl
[5] http://s3.amazonaws.com/ec2-downloads/ec2-api-tools.zip
[6] http://s3.amazonaws.com/ec2-downloads/CloudWatch-2009-05-15.zip
[7] http://s3.amazonaws.com/ec2-downloads/AutoScaling-2009-05-15.zip
[8] http://s3.amazonaws.com/ec2-downloads/ElasticLoadBalancing-2009-05-15.zip
[9] http://s3.amazonaws.com/rds-downloads/RDSCli.zip
[10] http://blog.rightscale.com/2009/10/05/amazon-usage-estimates/
[11] http://www.cloudstatus.com/