Holger Sirtl

(E-Mail: holger.sirtl@microsoft.com)
ist seit 2006 Architect Evangelist bei Microsoft. Schwerpunkte seiner Arbeit sind Softwarearchitekturen mit der Windows Azure-Plattform. Zuvor war er sechs Jahre für eine international führende Unternehmensberatung sowie zwei Jahre für einen großen deutschen Energieversorger tätig.

Eigene Services in der Cloud mit Windows Azure

Mit Windows Azure stellt Microsoft Entwicklern zahlreiche Werkzeuge zur Entwicklung eigener Cloud-basierter Anwendungen zur Verfügung. Neben einer Ausführungsumgebung für eigene Dienste und verschiedenen Speicherdiensten stellt Windows Azure auch Cloud Services bereit, mit denen Cloud-basierte Komponenten mit lokaler IT verknüpft werden können. Dabei können die Dienste von verschiedensten Technologien, wie .NET, PHP, Java etc. genutzt werden. Dieser Artikel zeigt, wie die Plattform aufgebaut ist und was bei der Entwicklung eigener Azure-basierter Anwendungen beachtet werden muss. Betrachtet wird dabei der gesamte Entwicklungsprozess angefangen beim Aufsetzen der Entwicklungsumgebung, über das Anlegen eines neuen Projekts, Entwicklung und Test bis hin zu Deployment und Betrieb in der Cloud.

Windows Azure – Platform-as-a-Service mit Microsoft

Cloud Computing-Dienste werden häufig entsprechend der Abstraktionsebene in Infrastruktur-, Platform- und Software-as-a-Service kategorisiert. Entsprechend dieser Einteilung lässt sich Windows Azure als Platform-as-a-Service-Angebot sehen, welches auch einige Elemente aus der Infrastrukturschicht enthält.

Windows Azure ist zum einen eine Sammlung von Cloud Services, die von Entwicklern zur Erstellung und dem Betrieb eigener Cloud-basierter Anwendungen genutzt werden können und die in Mi-

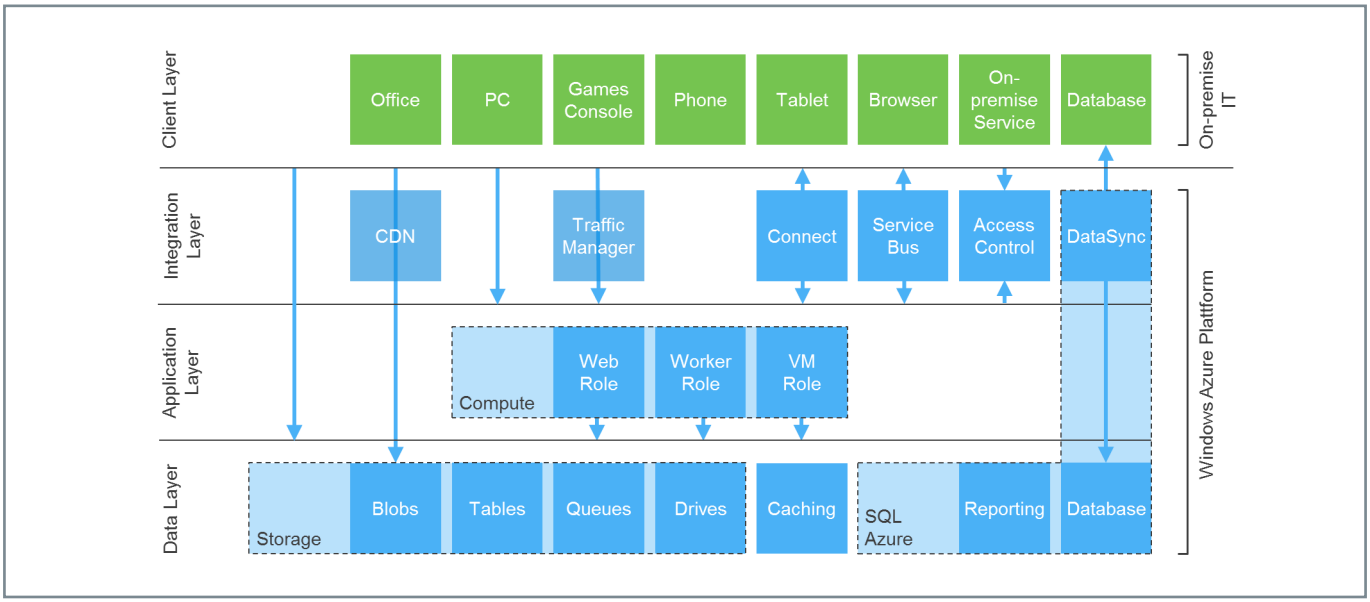

Abb. 1: Referenzarchitektur Azure-basierter Cloud-Anwendungen

crosofts weltweit verteilten Rechenzentren betrieben werden. Der Entwickler kann dabei den Ausführungsort eines jeden genutzten Azure Service bestimmen. Zum anderen bietet Windows Azure auch eine Reihe von Ressourcen, die Entwicklern bei Implementierung, Test und Deployment dieser Anwendungen helfen.

Anhand einer Referenzarchitektur, die in Abbildung 1 zu sehen ist, soll das Zusammenspiel der einzelnen Azure Services erläutert werden. Die Dienste lassen sich grob drei Schichten zuordnen:

- Der Datenschicht (*Data Layer*) gehören die Dienste an, über die Daten in der Cloud gespeichert und ausgewertet werden können.
- In der Anwendungsschicht (*Application Layer*) sind die Dienste, über die Programmcode in der Cloud ausgeführt werden kann.
- Die Dienste, die dem sicheren Datenaustausch und der Interaktion von Cloud-basierten Softwarekomponenten mit Komponenten der lokalen IT dienen, sind der Integrationsschicht (*Integration Layer*) zugeordnet.

Clients, die auf entsprechende Azure-basierte Dienste zugreifen, lassen sich einer Client-Schicht (*Client Layer*) zuordnen.

Die Windows Azure Storage Services auf der Datenschicht fassen alle Non-SQL-Persistenzdienste zusammen. Die Dienste sind über RESTful-Schnittstellen zugänglich. Daten werden automatisch dreifach innerhalb eines vom Entwickler wählbaren Rechenzentrums gespeichert. Eine automatische Replikation zu einem Rechenzentrum innerhalb einer geografischen Region kann aktiviert werden.

Wer für den Zugriff nicht den Weg über die RESTful-Schnittstellen gehen möchte, erhält in den frei verfügbaren Software Development Kits (SDKs) für .Net, Java, PHP und Node.JS entsprechende Klassenbibliotheken, in denen die RESTful-Aufrufe gekapselt werden.

Blob Storage dient zur Speicherung großer Binärdaten (Blob = binary large objects) wie Videos, Bilder, Musik etc. Das zuschaltbare Content Delivery Network kann Inhalte aus dem Blob Storage an weltweit verteilten Rechenzentrumsstandorten replizieren, vorhalten und Anwendern so sehr effizient ausliefern. Beim Zugriff erhält ein Anwender die Inhalte der betreffenden Blobs aus dem für ihn nächstgelegenen Rechenzentrum.

Table Storage stellt eine schemalose Non-SQL-Datenbank zur Speicherung semi-strukturierter Entity Sets zur Verfügung. Der Queue Storage ist ein Messaging Service, der dem Austausch kleiner Nachrichtenpakete dient. Windows Azure Drives stellen eine auf Blob Storage liegende Zugriffsschicht dar, die Clients eine NTFS-Dateisystemschnittstelle auf Blob Storage bereitstellt. Der Caching Service ist ein verteilter In-Memory-Cache, den Anwendungskomponenten zur temporären Zwischenspeicherung von Daten nutzen können.

Für ASP.NET sind Provider zur Ablage des Session-Zustands sowie Funktionen für das Page Output Caching für die effizientere Auslieferung von Webseiten verfügbar. Mit SQL Azure besitzt Windows Azure auch ein echtes relationales Datenbanksystem (RDBMS) als Cloud Service.

Wie beim Storage werden auch hier Daten dreifach gespeichert. SQL Azure bildet ein Subset von SQL Server ab und ist entsprechend schnittstellenkompatibel. Somit kann von allen Werkzeugen und Technologien, die mit SQL Server interagieren können, auch auf SQL Azure zugegriffen werden. Für .NET, Java, PHP können beispielsweise entsprechende SQL Server-Treiber direkt verwendet werden. Über den Reporting Service können Auswertungen über SQL Azure-Datenbestände erstellt und Anwendungen zur Anzeige zugänglich gemacht werden.

Der DataSync Service erlaubt darüber hinaus die Synchronisation in SQL Azure gespeicherte Datenelemente zwischen SQL Azure und lokal betriebenen SQL Server Datenbanken. Dabei können Synchronisationsrichtung, Konfliktauflösungsstrategie und die zu synchronisierenden Elemente (ganze Datenbanken, einzelne Tabellen, bestimmte Spalten und Records) konfiguriert werden.

Der Windows Azure Compute Service auf der Anwendungsschicht ermöglicht die Ausführung eigener Anwendungen in der Cloud. Diese können aus mehreren Komponenten aufgebaut sein, die im Kontext von Azure als Rollen bezeichnet werden. Anwendungen können mithilfe der in den SDKs enthaltenen Werkzeugen paketiert und in den Compute Service geladen werden. Der Prozess ist in Abbildung 2 zu sehen.

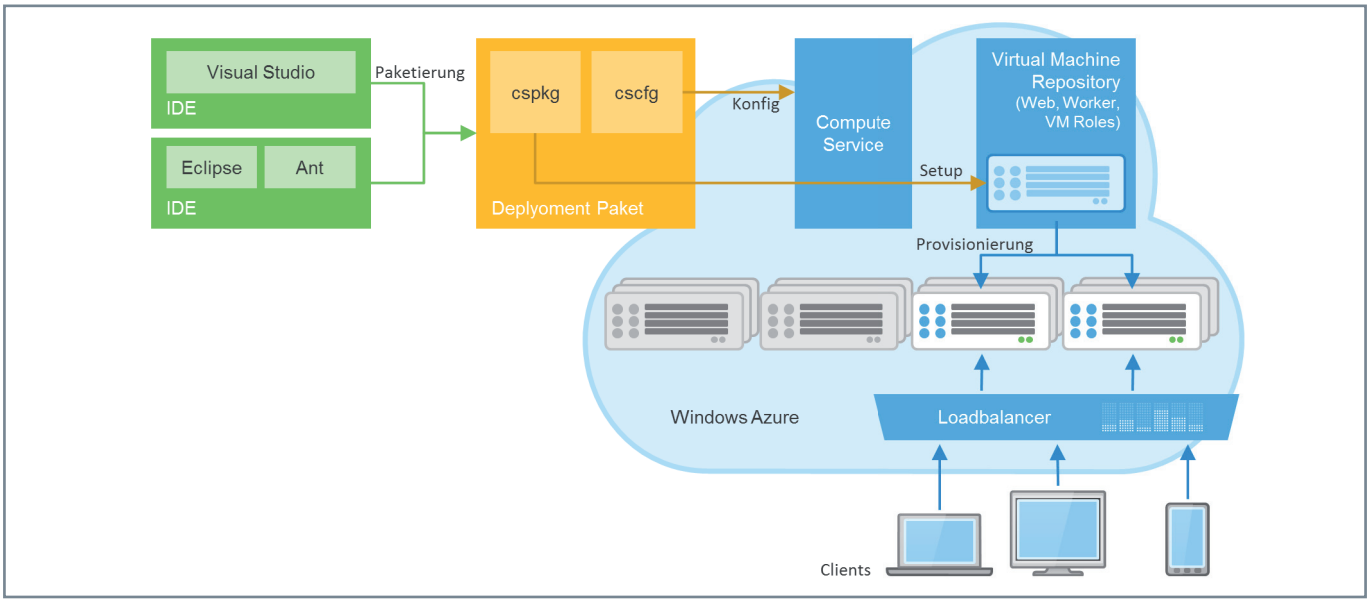

Abb. 2: Prozess zur Paketierung und Provisionierung eigener Cloud Services

Die Bereitstellung von virtuellen Maschinen (VMs) in der gewünschten Anzahl und Größe, deren Konfiguration, die Installation des Anwendungspaketes, die Einstellung eines vorgeschalteten Loadbalancers etc. wird vollautomatisch durch den Compute Service gesteuert. Der Entwickler muss sich demnach nicht mit dem Setup virtueller Maschinen auseinandersetzen. Diese werden aus einem Pool bereits vorkonfigurierter VMs bezogen.

Dabei gibt es drei Typen von VMs. Der Entwickler bestimmt, welche Rollen (und damit welche VM-Vorlagen) zum Einsatz kommen und wie viele VM-Instanzen jeweils für einzelne Rollen bereitgestellt werden. Die VMs von Web Roles enthalten einen vorinstallierten Internet Information Server (IIS). Web Roles eignen sich demnach für die Implementierung ASP.NET-basierter Webanwendungen bzw. -services.

Worker Role VMs unterscheiden sich von Web Roles darin, dass der IIS fehlt. Worker Roles sind in Fällen sinnvoll, in denen Hintergrundprozesse ausgeführt oder ein anderer Webserver als IIS eingesetzt werden soll (beispielsweise Apache Webserver oder Tomcat).

Bei VM Roles kann das VM Image von Entwicklern selbst erstellt und in Azure betrieben werden. Auf alle VMs hat der Entwickler vollen Administratorzugang. Auch die Möglichkeit des Remote Desktop-Zugangs auf einzelne VM Instanzen besteht.

Cloud-basierte Anwendungen haben in der Regel mindestens eine Komponente, die in der lokalen IT ausgeführt wird. Im einfachsten Fall ist dies (bei Webanwendungen) ein HTML-Frontend, welches im lokalen Browser ausgeführt wird. Aber auch komplexere Anwendungsteile, Rich Clients, Office Add-Ins, Datenbanken etc. sind denkbar. Tatsächlich tragen Hybride-Szenarien, in denen Teile einer Anwendung lokal und andere in der Cloud ausgeführt werden, dem Wunsch Rechnung, die Vorteile der Cloud (hohe Skalierbarkeit, flexible Verfügbarkeit, nutzungsabhängige Kosten etc.) mit den Stärken der lokalen IT (individuelle Konfigurationen, Compliance-Anforderungen etc.) zu kombinieren.

Auf der Integrationsschicht (siehe Abbildung 1) bietet Windows Azure deshalb eine Reihe von Cloud Services, die Funktionen für Interaktion und Integration von lokaler- mit Cloud-IT bereitstellen. Das Content Delivery Network (CDN) erlaubt, Inhalte aus dem Blob Storage in weltweit verteilten Rechenzentren zu cachen und so Clients effizient bereitzustellen. Der Traffic Manager stellt eine ähnliche Funktion für Anwendungslogik zur Verfügung, indem er Zugriffe auf Cloud Services, für die Instanzen in verschiedenen Rechenzentren verteilt sind, entsprechend vom Entwickler konfigurierbarer Richtlinien auf einzelne Instanzen leitet. CDN und Traffic Manager optimieren Daten- und Anwendungszugriffe, die von der lokalen IT initiiert werden.

Soll von einem Azure Service heraus eine Ressource in der lokalen IT angesprochen werden, bietet Windows Azure Möglichkeiten auf mehreren Ebenen:

- Windows Azure Connect ermöglicht auf *Netzwerkebene* den Aufbau einer IPSec-gesicherten Verbindung zwischen Rollen in Windows Azure und Rechnern in der lokalen IT.
- Auf *Anwendungsebene* bietet der Windows Azure Service Bus die Möglichkeit, lokale Services untereinander bzw. auch mit Azure Services zu vernetzen. Lokale Services, die sich beim Service Bus registrieren, können so von Azure aus aufgerufen werden. Die Service Bus-Infrastruktur kann über entsprechende Software Development Kits (SDKs) aus verschiedenen Technologien wie .NET, Java, PHP etc. heraus genutzt werden.
- Auf *Datenebene* kann der schon genannte SQL Azure DataSync Service verwendet werden, um Daten zwischen lokalen und Azure-basierten Datenbanken zu synchronisieren.

Der Windows Azure Access Control Service kann in seiner Funktion als Cloud-basierter Security Token Service verschiedene externe Identity Provider, z. B. Windows Live ID, Google ID, Yahoo ID, Facebook ID oder Active Directory Federation Services (ADFS) 2.0 eigenen Anwendungen zugänglich machen. Diese haben damit die Möglichkeit, eigene Benutzer über ebendiese Identity Provider zu authentifizieren.

Entwicklung eigener Azure-basierter Anwendungen

Entwickler können nach Bedarf einzelne Dienste der Windows Azure-Plattform auswählen und nutzen. Sie benötigen hierzu einen Account, über den die Dienstnutzung abgerechnet wird. Neben den regulären, kostenpflichtigen Zugängen, bei denen die Services nutzungsabhängig oder bei Bedarf im Rahmen eines Abomodells abgerechnet werden (auch Kombinationen aus Abo- und Nutzungsabrechnung sind möglich), besteht auch die Möglichkeit, Azure 90-Tage lang kostenfrei zu testen (Details siehe [90-T]).

Der Entwicklungsprozess selbst ist dem einer klassischen Software sehr ähnlich. Dieser ist in Abbildung 3 zu sehen. Auf dem Windows Azure Portal [WAP] findet der Entwickler SDKs für .NET, Java, PHP und Node.JS. Das .NET-SDK enthält Projektvorlagen für Visual Studio, Build- und Deploymentwerkzeuge sowie Klassenbibliotheken zur Nutzung aller Windows Azure Services.

Im Java-SDK befinden sich Projektvorlagen für Eclipse, ein Build-Skript für Ant sowie Klassenbibliotheken für die wichtigsten Services. PHP-Entwickler erhalten Projektvorlagen, Kommandozeilen-Skripte sowie ebenfalls Klassenbibliotheken. Entwickler können demnach weiter mit den von ihnen gewohnten Werkzeugen arbeiten.

Neben den SDKs stehen auf dem Portal auch Emulator-Packages zur Verfügung. Mit diesen Emulatoren können der Windows Azure Compute und die Storage Services sowie SQL Azure lokal nachgebildet und damit Cloud Services in der Entwicklungsumgebung getestet werden. Voraussetzung für den Compute Emulator ist ein lokal installierter IIS, für die Storage Services ein lokaler SQL Server (auch die kostenlose Express Edition ist möglich).

Nach dem Aufsetzen der Entwicklungsumgebung kann die Entwicklung beginnen. Nachdem in den Windows Azure VMs Windows Server als Gastbetriebssystem zum Einsatz kommt, kann auf Windows Azure grundsätzlich alles ausgeführt werden, was auch auf einer entsprechenden lokalen Umgebung möglich ist. Entwickler können in den von den SDKs vorkonfigurierten Projekten beliebig weitere Frameworks, z. B. Dynamische Verbindungsbibliotheken nutzen. Sie müssen lediglich sicherstellen, dass diese dann auch in das Deployment-Paket aufgenommen werden und etwaige Setup-Routinen in sogenannten Startup Tasks beim Hochfahren der VMs automatisiert ausgeführt werden.

Bevor Cloud Services tatsächlich in der Cloud ausgeführt werden, können diese in der lokalen Emulationsumgebung getestet

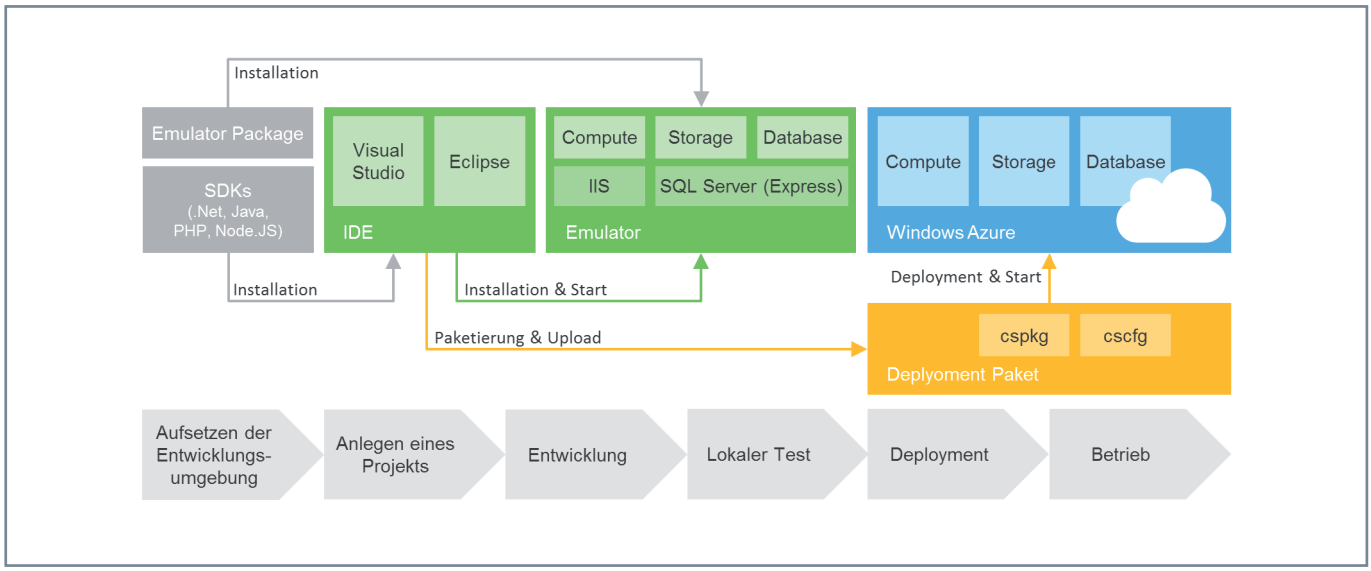

Abb. 3: Prozess zur Entwicklung eines Azure-basierten Cloud Service

werden. Dabei stehen die bekannten Möglichkeiten zum Debugging und zur Überwachung quasi unverändert zur Verfügung. Der Compute Emulator ermöglicht auch die Ausführung von Rollen in mehreren Instanzen und eine entsprechende Lastverteilung zwischen diesen Instanzen. Somit können auch Multiinstanzinstallationen getestet werden.

Nach erfolgreichem lokalen Test können die fertigen Anwendungen dann mithilfe der in den SDKs enthaltenen Werkzeugen in ein Deployment-Paket geschnürt, dieses in den Azure Account geladen und dort gestartet werden. Die weiteren Schritte zur Konfiguration der benötigten VMs (in der gewünschten Größe und Anzahl), des Azure Loadbalancers etc. übernimmt Windows Azure.

Im laufenden Betrieb stellt Windows Azure automatisch sicher, dass immer die gewünschte Zahl an Instanzen ausgeführt wird. Erkennt Windows Azure den Ausfall einer Instanz, wird sofort eine neue gestartet. Darüber hinaus etabliert es Sicherheitsmechanismen (Firewall-Konfiguration, verschlüsselte Datenübertragung, Abwehr von DoS-Attacken etc.), die auf Infrastruktur- und Plattformebene einen ordnungsgemäßen Betrieb sicherstellen.

Auf Anwendungsebene kann eine Cloud-Anwendung auf verschiedenen Ebenen überwacht und gesteuert werden. Zunächst ist es möglich, sich mit einer VM-Instanz per Remote Desktop zu verbinden und den Status (z. B. per Task Manager) zu analysieren. Einen ganzheitlichen Blick auf den Status einer Anwendung (d. h. auf alle Rollen und Instanzen) ermöglicht die Diagnostics API. Über diese können alle Performance- und Statusindikatoren, also Logdateien, Performance Counter etc. in konfigurierbaren zeitlichen Abständen in Windows Azure Storage übertragen und von dort ausgelesen werden. Dies ermöglicht eine laufende Überwachung des Anwendungsstatus.

Scale-Out-Paradigma

Mithilfe dieser Diagnosedaten kann das Setup einer Cloud-Anwendung gesteuert werden. So kann zur Laufzeit ohne Umgruppierung die Zahl der Instanzen geändert werden. Dies kann bei schwankendem Lastverlauf sinnvoll sein, um zu jedem Zeitpunkt nur so viele Instanzen zu betreiben, wie für die anliegende Last erforderlich ist. Anwendungen müssen auf diese Form der Skalierung, bei der nicht die Größe vorhandener Instanzen (Scale-up) sondern die Zahl der Instanzen (Scale-out) geändert wird, vorbereitet sein.

Cloud-Anwendungen sollten deshalb grundsätzlich möglichst zustandslos entworfen werden. Der Session-Zustand sollte nicht in einer einzelnen Instanz gehalten, sondern über alle Instanzen (z. B. über Persistierung in Windows Azure Storage oder die Nutzung des Windows Azure Caching Service) geteilt werden.

Fazit

Windows Azure bietet Entwicklern verschiedener Technologien (.NET, Java, PHP etc.) die Möglichkeit, eigene Cloud Services zu schreiben, zu testen und in der Cloud zu betreiben und zu steuern. Aber auch lokal betriebene klassische Anwendungen können punktuell einzelne Cloud Services (z. B. Storage, Datenbank) nutzen. Diese stehen flexibel zur Verfügung und werden nutzungsabhängig abgerechnet.

Auch die Kombination lokaler Anwendungs- mit Cloud-Komponenten wird von Windows Azure auf verschiedenen Ebenen mit entsprechenden Cloud Services unterstützt. Damit ergeben sich für Entwickler neue Möglichkeiten, den optimalen Ausführungsort einzelner Komponenten zu bestimmen und leistungsfähige Anwendungssysteme zu implementieren.

Über einen kostenfreien Testaccount können die Services (mengenmäßig begrenzt, funktional aber uneingeschränkt) unverbindlich getestet werden. ■

Referenzen

[90-T] 90-Tage-Testaccount für Windows Azure
http://www.windowsazure.com/de-de/pricing/free-trial/
[WAP] Windows Azure Portal
http://www.azure.com/